Report2

February 21, 2021

1 Report 2

Authors: Richard Liang, Olivia majedi, Priyanka Kishore, Rhea Rakheja, Jiadong Li, Michael Strobel

[1]:
```
import pandas as pd
```

[2]:
```
df = pd.read_csv('https://raw.githubusercontent.com/jdli28/STAT440/master/
↪summer_mount_ginini.csv')
```

[3]:
```
df = df.dropna()
```

[4]:
```
df
```

[4]:
```
          Date     Location  MinTemp  MaxTemp
0    12/1/2008  MountGinini      5.2     13.0
1    12/2/2008  MountGinini      3.0     15.0
2    12/3/2008  MountGinini      6.0     15.0
3    12/4/2008  MountGinini      2.0     15.0
4    12/5/2008  MountGinini      8.0     17.8
..         …            …       …       …
743  2/24/2017  MountGinini     13.8     24.1
744  2/25/2017  MountGinini      9.5     11.2
745  2/26/2017  MountGinini      4.7     16.7
746  2/27/2017  MountGinini      5.6     16.2
747  2/28/2017  MountGinini      7.5     18.0

[745 rows x 4 columns]
```

[5]:
```
import numpy as np
```

1.1 Population size N and true parameter mu(MaxTemp)

[6]:
```
N = len(df)
N
```

[6]: 745

[7]:
```
max_temp_mean = np.mean(df.MaxTemp)
max_temp_mean
```

[7]: 19.112885906040272

1.2 Calculate sample size n for 90% and 95% confidence levels and couple different d's. Use true ^2 for these calculations

[8]:
```
sigma_sq = np.var(df.MaxTemp)
sigma_sq
```

[8]: 21.75814267825774

[9]:
```
r = [.05, .01, .1]
d = [(max_temp_mean * rval) for rval in r]
d
```

[9]: [0.9556442953020137, 0.19112885906040272, 1.9112885906040273]

[10]:
```
 from scipy import stats
z_alpha_90 = stats.norm.ppf(1-0.05)
z_alpha_95 = stats.norm.ppf(1-0.025)
```

[11]:
```
n_90 = []
n_95 = []
```

[12]:
```
for d_val in d:
  n0 = z_alpha_90**2*sigma_sq/(d_val**2)
  n_90.append(
    1/((1/n0)+(1/N))
  )

  n0 = z_alpha_95**2*sigma_sq/(d_val**2)
  n_95.append(
    1/((1/n0)+(1/N))
  )
```

[13]:
```
import math
```

[14]:
```
n_90
n_90 = [math.ceil(n) for n in n_90]
n_90
```

[14]: [60, 510, 16]

[15]:
```
n_95
n_95 = [math.ceil(n) for n in n_95]
n_95
```

[15]: [82, 563, 23]

1.3 Estimate your parameter of interest using SRS with n's which you calculated above.

1.3.1 90% CI

[16]:
```
sample_90s = []
sample_95s = []
```

[17]:
```
for n in n_90:
  sample = np.random.choice(df.MaxTemp, size=n, replace=False)
  sample_90s.append(sample)
  print(np.mean(sample))
```

```
19.108333333333334
19.12117647058824
20.25625
```

1.3.2 95% CI

[18]:
```
for n in n_95:
  sample = np.random.choice(df.MaxTemp, size=n, replace=False)
  sample_95s.append(sample)
  print(np.mean(sample))
```

```
19.954878048780486
19.223445825932505
17.352173913043476
```

1.4 Estimate variance of your estimator for these n's

[19]:
```
for sample in sample_90s:
  print(np.var(sample))
```

```
17.410763888888887
22.90453194925029
9.679960937499999
```

[20]:
```
for sample in sample_95s:
  print(np.var(sample))
```

```
28.51686644854253
21.62307373907227
18.305973534971645
```

1.5 Calculate confidence intervals for these estimators.

1.5.1 90% CI for ybar

[21]:
```
def get_ybar_CI(ybar, z, n, s_sq):
  upper = ybar + z * np.sqrt(
    ((N-n)/N)*(s_sq/n)
  )
  lower = ybar - z * np.sqrt(
    ((N-n)/N)*(s_sq/n)
  )
  return [lower, upper]
```

[22]:
```
for sample in sample_90s:
  ybar = np.mean(sample)
  n = len(sample)
  s_sq = np.var(sample)
  ci = get_ybar_CI(ybar, z_alpha_90, n, s_sq)
  print(ci)
```

```
[18.25870752919851, 19.95795913746816]
[18.925400820732197, 19.31695212044428]
[18.990669730177896, 21.521830269822107]
```

1.5.2 95% CI for ybar

[23]:
```
for sample in sample_95s:
  ybar = np.mean(sample)
  n = len(sample)
  s_sq = np.var(sample)
  ci = get_ybar_CI(ybar, z_alpha_95, n, s_sq)
  print(ci)
```

```
[18.864516084072232, 21.04524001348874]
[19.033596016347715, 19.413295635517294]
[15.630816514537186, 19.073531311549768]
```

1.6 Choosing optimal sample sizes

[24]:
```
n = n_95[0]
n
```

[24]: 82

1.7 Guaranteeing the nominal confidence level

[25]:
```
d_val = 0.9556442953020137
```

[26]:
```
ct = 0
for i in range(100):
  sample = np.random.choice(df.MaxTemp, size=n, replace=False)
  ybar = np.mean(sample)
  d = abs(ybar - max_temp_mean)
  print(d)
  if d - d_val < 0:
    ct +=1
print("----------")
print(ct)
```

```
0.4970322475036859
0.7702029792110068
0.22491897200850985
0.06882141103289996
0.21760189883777414
0.3677639548207594
0.5188214110328992
0.0700409232280208
0.5663823866426512
0.13849566213783504
0.10784580127679888
0.5287395645768562
0.262885906040274б
0.39337371091832196
0.08833360615484764
1.4968701915207028
0.2494712718939276
0.08711409395973035
0.7458127353085651
0.572479947618266
0.09581273530856649
0.35556883286954033
0.29443116713045825
0.0029677524963105384
```

0.12613848420363283
0.990772630545095
0.43101653298412046
0.05434932067441878
0.17508102799149228
0.8078458012768053
0.18239810116222444
1.0458127353085658
0.6823981011622209
0.6224799476182632
0.11532493043051417
0.042154198723196146
0.16516287444753175
0.7493092159109445
0.3517482403011911
0.5958127353085665
0.5006907840890555
0.757845801276801
0.1993092159109473
0.45191029628417567
0.2761384842036314
0.8066262890816809
0.11882141103289712
0.2970322475036866
0.4884956621378329
0.1749189720085056
0.37873956457685765
0.009227369454904988
0.28101653298411833
0.36776395482076296
1.1651628744475317
0.28239810116222586
0.1726420036012506
0.34947127189393257
0.18605663774759051
0.7567883450646633
0.24808970371582717
0.028739564576856225
0.5066262890816766
0.2665444426256407
0.11882141103289712
0.14337371091832196
0.9310165329841205
0.5555688328695396
0.15312980847929936
0.6702029792110018
0.4531298084792965
0.1445932231134428

```
0.02264200360124846
0.8836176133573446
0.507845801276801
0.45922736945490783
0.43711409395972467
0.5005287281060653
0.6102848256670441
0.37264200360124633
0.5458127353085658
0.6310165329841162
1.018983467015886
0.594431167130459
0.012885906040271067
0.1895531183499699
0.8689834670158838
0.3031298084792944
0.07630054018661525
0.6176018988377727
0.14703224750368804
0.1310165329841162
0.16638238664264904
0.03727614994271278
0.14947127189392972
0.46150433786216993
0.4456506793255848
0.21654444262564
0.03849566213783362
0.3651628744475346
----------
95
```

Exactly 95/100 of our samples have differences less than d=0.9556442953020137

```
[ ]:
```