CSCI3022 S22

Homework 5: Confidence Intervals

Due Monday, April 4 at 11:59 pm to Canvas and Gradescope

Name: Matthew Su

Your solutions to computational questions should include any specified Python code and results as well as written commentary on your conclusions. Remember that you are encouraged to discuss the problems with your classmates, but **you must write all code and solutions on your own.**

NOTES:

- Any relevant data sets should be available on Canvas. To make life easier on the graders if they need to run your code, do not change the relative path names here. Instead, move the files around on your computer.
- If you're not familiar with typesetting math directly into Markdown then by all means, do your work on paper first and then typeset it later. Here is a reference guide linked on Canvas on writing math in Markdown. **All** of your written commentary, justifications and mathematical work should be in Markdown. I also recommend the wikibook for LaTex.
- Because you can technically evaluate notebook cells is a non-linear order, it's a good idea to do **Kernel** $\rightarrow$ **Restart & Run All** as a check before submitting your solutions. That way if we need to run your code you will know that it will work as expected.
- It is **bad form** to make your reader interpret numerical output from your code. If a question asks you to compute some value from the data you should show your code output **AND** write a summary of the results in Markdown directly below your code.
- 45 points of this assignment are in problems. The remaining 5 are for neatness, style, and overall exposition of both code and text.
- This probably goes without saying, but... For any question that asks you to calculate something, you **must show all work and justify your answers to receive credit**. Sparse or nonexistent work will receive sparse or nonexistent credit.
- There is *not a prescribed API* for these problems. You may answer coding questions with whatever syntax or object typing you deem fit. Your evaluation will primarily live in the clarity of how well you present your final results, so don't skip over any interpretations! Your code should still be commented and readable to ensure you followed the given course algorithm.
- There are two ways to quickly make a .pdf out of this notebook for Gradescope submission. Either:

- Use File -> Download as PDF via LaTeX. This will require your system path find a working install of a TeX compiler
- Easier: Use File -> Print Preview, and then Right-Click -> Print using your default browser and "Print to PDF"

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pylab as plt
import scipy.stats as stats
%matplotlib inline
```

Shortcuts: Problem 1 | Problem 2 | Problem 3 |

Back to top

(10 pts) Problem 1 (Computation): How did we do?

In prior (and future) homeworks and the exam coding, we have placed a large emphasis on using **simulations** to estimate probabilities. In these cases, we're actually simulating a process or game and then counting outcomes as *proportions* to estimate what we believe are the exact *probabilities*.

On Homework 3 we we just stated that our simulated *proportions* would match the theoretical *probabilities* if our sample size was large enough. But it's important in data science to always convey our sense of error, in some form of **uncertainty quantification**. Confidence intervals are a simple way to do this!

A: For *your* results from the "poker hands" problem of HW 3 Problem 2, what is the corresponding 99% confidence interval on the probability of getting 3-of-a-kind?

Note: If you didn't actually do this problem, you may use the solution simulation values of 213 "3-of-a-kind" hands in 10000 simulations. But use your own exact value if possible!

In [4]:
```
mean = 213/10000
zcrit=stats.norm.ppf(.99)
var_t=mean*(1-mean)/10000
print("99% CI of: ", mean-zcrit*np.sqrt(var_t),   mean+zcrit*np.sqrt(var_t))
```

```
99% CI of:  0.0179411604831872 0.0246588395168128
```

B: For *your* results from the "NFL overtime" problem of HW 3 Problem 3, what is the 99% corresponding confidence interval on how much more likely it is for team 1 (receiving the ball first) to win than team 2?

Note: If you didn't actually do this problem, you may use the solution simulation values of 542 "team 1 wins" and 321 "team 2 wins" in 1000 simulations. But use your own exact values if possible!

In [111…

```
# Win prob for team 1 used

mean = 0.649
zcrit=stats.norm.ppf(.99)
var_t=mean*(1-mean)/10000

# Win prob for team 2 used

mean2 = 0.202
zcrit2=stats.norm.ppf(.99)
var_t2=mean2*(1-mean2)/10000

# print("For team1, a 99% CI of: ", mean-zcrit*np.sqrt(var_t),   mean+zcrit*np.sqrt(var_
# print("For team2, a 99% CI of: ", mean2-zcrit2*np.sqrt(var_t2),   mean2+zcrit2*np.sqr

print("CI: ", mean-mean2 - zcrit*np.sqrt(var_t+var_t2), mean-mean2 + zcrit*np.sqrt(var_
```

```
CI:  0.43249069307579446 0.46150930692420555
```

Back to top

(10 pts) Problem 2 (Theory): How big a Sample?

As Problem 1 suggests, the goal of our simulations is usually to be "as close as possible" to the true value, but we may need pretty large numbers of simulations to get narrower confidence intervals than the above!

Towards that end, use Markdown to answer the following **exactly**:

How many simulations are necessary to ensure that the confidence interval centered around a sample proportion $\hat{p}$ is of width **at most** 1%? Use a significance of 95\% and approximate that $z_{0.025} = 2$.

Hint: The width of the confidence interval for proportions depends on $\hat{p}$. What is the worst case scenario that makes it widest? How big should n be to account for this?

$n = \frac{2^2 * 0.5(1-0.5)}{0.01^2} = 10000$

Back to top

(25 pts) Problem 3 (Computation): How different are the months?

Consider the Boulder Weather data also used in Homework 2.

Load in the data in boulder_daily_weather.csv

Information on the file is as follows:

- Temperatures are in degrees Fahrenheit and snow and precipitation are in inches.

- Precipitation is the total liquid equivalent of all forms of precipitation (rain, snow, hail,...).
- Missing values are indicated by -998.0.
- Trace values (less than 1/10 for snow and 1/100 for rain) are indicated by -999.0.
- Your file starts in 1970, although limited data goes back to 1899 from NOAA.

In [3]:
```
df=pd.read_csv('../data/boulder_daily_weather.csv', encoding='UTF-8')
print(df.head(2))
print(df.dtypes)
df.head(5)
df.describe()
```

```
   Year  Month  Day  MaxT  MinT  Precip  Snow  SnowDepth
0  1970      1    1    32    19  -999.0  -999          3
1  1970      1    2    33    15     0.0     0          3
Year           int64
Month          int64
Day            int64
MaxT           int64
MinT           int64
Precip       float64
Snow           int64
SnowDepth      int64
dtype: object
```

Out[3]:

	Year	Month	Day	MaxT	MinT	Precip	Snow
count	19032.000000	19032.000000	19032.000000	19032.000000	19032.000000	19032.000000	19032.00000
mean	1995.500000	6.513661	15.756831	38.490280	11.295607	-149.310692	-85.29235
std	15.008725	3.451325	8.811761	167.879418	164.444183	356.237912	279.50044
min	1970.000000	1.000000	1.000000	-998.000000	-999.000000	-999.000000	-999.00000
25%	1982.750000	4.000000	8.000000	51.000000	26.000000	0.000000	0.00000
50%	1995.500000	7.000000	16.000000	65.000000	38.000000	0.000000	0.00000
75%	2008.250000	10.000000	23.000000	80.000000	51.000000	0.010000	0.00000
max	2021.000000	12.000000	31.000000	102.000000	77.000000	9.080000	22.00000

A: Remove all locations where the `MaxT` variable is set to the missing or trace fill value. Do **not** remove any rows that have `MaxT` but are missing other variables.

In [9]:
```
clean = df.loc[(df["MaxT"] > -998)]
# print(clean.count())
clean.head()
```

Out[9]:

	Year	Month	Day	MaxT	MinT	Precip	Snow	SnowDepth
0	1970	1	1	32	19	-999.0	-999	3
1	1970	1	2	33	15	0.0	0	3
2	1970	1	3	29	14	0.0	0	2
3	1970	1	4	33	9	0.0	0	2

	Year	Month	Day	MaxT	MinT	Precip	Snow	SnowDepth
4	1970	1	5	18	3	0.1	2	4

B: For each of the 12 months of the year, find and print the means, standard deviations, and number of observations of the `MaxT` variable for that month.

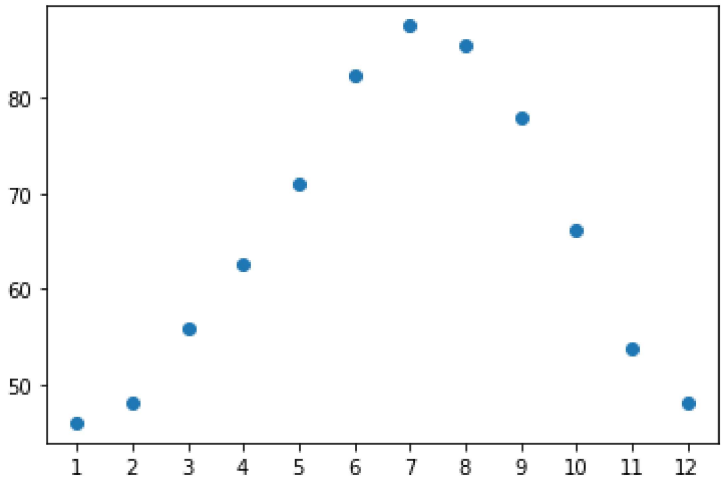
In [33]:
```
print("STD is: ", clean['MaxT'].std())
print("Mean is: ", clean['MaxT'].mean())
print("Number of Observations: ", clean['MaxT'].count())

# clean["MaxT"].describe()
```

```
STD is:  18.30799297687756
Mean is:  65.36493989542343
Number of Observations:  18551
```

Out[33]:
```
count    18551.000000
mean        65.364940
std         18.307993
min        -12.000000
25%         52.000000
50%         66.000000
75%         81.000000
max        102.000000
Name: MaxT, dtype: float64
```

C: Make a plot where the X-axis is the month of the year (1-12) and the y-axis is the mean `MaxT`. Do any of the months appear very close in temperature?

In [75]:
```
# clean["MaxT"].plot(column="Month")
cleanp = clean[["MaxT", "Month"]]

# print(cleanp.loc[cleanp["Month"] == i, "MaxT"].mean())

MaxtM = np.array([0]*12, dtype=float)
for i in range(1,12):
    MaxtM[i-1] = cleanp.loc[cleanp["Month"] == i, "MaxT"].mean()

MaxtM[11] = cleanp.loc[cleanp["Month"] == 2, "MaxT"].mean()

# print(MaxtM)

months = np.array([1,2,3,4,5,6,7,8,9,10,11,12])

plt.scatter(months, MaxtM)
plt.xticks(months)

plt.show()


# cleanp["MaxTm"] = cleanp["MaxT"].mean()
# plotdata.plot.scatter(x = "Month", y= 'MaxTm')
```



Generally speaking, Febuary and December share similar average tempuratrues, as well as March and November roughly speaking. April and October also seem to be within range of each other, and July and Augest are close as well. It very much resembles a quadratic in how it rises and falls in tempurature, with July being the peak of this quadratic.

D: For each *pair* of months (a total of $\binom{n}{2}$ pairs), compute a 99% confidence interval on the difference in the mean maximum temperatutes `MaxT`. Use the large-sample normal approximation.

Whenever 0 is inside the confidence interval for that difference in means, make note of it.

In [109…

```
# Based on: https://www.dummies.com/article/academics-the-arts/math/statistics/creating
# (Cause I'm a dummy)

for j in range(1,12):
    for i in range (1,12):
    # sample 1/2 mean (per month)
        if (i == j):
            continue
        else:
            x1 = cleanp.loc[cleanp["Month"] == j, "MaxT"].mean()
            x2 = cleanp.loc[cleanp["Month"] == i, "MaxT"].mean()
            # sample std
            s1 = cleanp.loc[cleanp["Month"] == j, "MaxT"].std()
            s2 = cleanp.loc[cleanp["Month"] == i, "MaxT"].std()
            # sample size of each month
            size1 = cleanp.loc[cleanp["Month"] == j, "MaxT"].count()
            size2 = cleanp.loc[cleanp["Month"] == i, "MaxT"].count()

            # pooled = ((size1 - 1)*(s1**2) + (size2-1)*(s2**2)/size1+size2-2)**2

            # comp = (x1-x2) + 2.76*(np.sqrt((pooled/size1) + (pooled/size2)))
            # compm = (x1-x2) - 2.76*(np.sqrt((pooled/size1) + (pooled/size2)))

            cplus = (x1-x2) + 2.76 * (np.sqrt((s1**2/size1) + (s2**2/size2)))
            cminus = (x1-x2) + 2.76 * (np.sqrt((s1**2/size1) + (s2**2/size2)))


            # Not 100% sure what the question is asking, more or less. I dont think any
            if ((abs(cplus) < 1) | (abs(cminus) < 1)):
                print(j, ",", i, cplus,cminus)
```

```
            # if ((round(cplus) == 0) | (round(cminus) == 1)):
            #     print(j, ",", i, cplus,cminus)
```

```
1 , 2 -0.8736409200952258 -0.8736409200952258
11 , 3 -0.9375376943499658 -0.9375376943499658
```

E: What months have "about the same" *average* maximum temperature in part **D**? Does this match what you expected from part **C**?

It looks like, roughly speaking (since I was using float), it appears that my March/November prediction was spot on, and it turns out that January and Febuary seem to have close average maximum temps as well.

F: In the prior problem, we computed a lot of intervals, and we allowed each interval to have a confidence of 99%. What is the probability that *each and every* interval comparing months A and B contained the true value of $\mu_A - \mu_B$?

Hint: this is not the same question as follows in Part G!

It's based on the confidence interval that we calculated.

G: Suppose we are *planning* to gather a large data set, where we will construct $\binom{n}{2}$ independent confidence intervals for the respective means μ_i of $\binom{n}{2}$ independent random samples, each of which is constructed at a 99% confidence.

What is the probability that *each and every* interval ends up convering the true mean for that feature/variable?