MidTerm2-TakeHome-Deploy

March 30, 2021

0.1 Full name:

0.2 R#:

0.3 HEX:

0.4 Exam 2 Take-Home_{your name}

0.5 Date:

1 Problem 1 (1 pts.)

Run the cell below as-is! It should produce no apparent output, but will make Markdown tables left align.

[7]:
```
%%html
<!--Script block to left align Markdown Tables-->
<style>
  table {margin-left: 0 !important;}
</style>
```

```
<IPython.core.display.HTML object>
```

2 Problem 2 (1 pts.)

Run the cell below as-is! If it produces an error, ignore and continue; dont try to fix the error

[8]:
```
# Preamble script block to identify host, user, and kernel
import sys
! hostname
! whoami
print(sys.executable)
```

```
atomickitty.aws
engr1330content
/opt/jupyterhub/bin/python3
```

---

3 Problem 2 (8 pts)

The table below contains some experimental observations.

Elapsed Time (s)	Speed (m/s)
0	0
1.0	3
2.0	7
3.0	12
4.0	20
5.0	30
6.0	45.6
7.0	60.3
8.0	77.7
9.0	97.3
10.0	121.1

1. Plot the speed vs time (speed on y-axis, time on x-axis) using a scatter plot. Use blue markers.
2. Plot a red line on the scatterplot based on the linear model $f(x) = mx + b$
3. By trial-and-error find values of m and b that provide a good visual fit (i.e. makes the red line explain the blue markers).
4. Using this data model estimate the speed at $t = 15$ `sec`.

3.1 Deliverables:

- Working scripts that produce requesite plots
- Narrative (or print blocks) that supply answer questions
- CCMR citations for sources (Lab 14, Lesson 9, ... is OK for using internal sources, URL for outside sources)

3.2 Hints:

- A suggested set of code cells is listed below
- Add/remove cells as needed for your solution

[9]:
```
# Create two observation lists; time  and speed
```

[10]:
```
# Create a data model function
```

[11]:
```
# Create a model speed list - using observation time list
```

[12]:
```
# Create a point plot of observed time and speed, overlay a line plot of model␣
↪time and speed
```

[13]:
```
# Report best values m and b
```

[14]:
```
# Estimate speed@ t = 15 sec. using fitted model
```

---

4 Problem 2 (15 pts)

Consider the script below, which crudely implements a simulation of Russian Roulette. How many times can you spin the cylinder and pull the trigger, before you fail? Play the game 25 times, record the pull count until failure.

1. Create a list of pulls until failure for each of your 25 attempts
2. Make a histogram of the list.
3. From your list, estimate the mean number of pulls until failure.
4. Plot the estimate as a red dot on the histogram

In the movie `The Deer Hunter` https://en.wikipedia.org/wiki/The_Deer_Hunter the captured soldiers modify the Russian Roulette Game by using more than a single cartridge.

3. Modify the program to the number of cartridges in the movie (3) and play again 25 times, record your pulls to failure
4. Make a second histogram of the `Deer Hunter` version of the game.
5. From your list, estimate the mean number of pulls until failure under the `Deer Hunter` conditions.
6. Plot the estimate as a red dot on the histogram
7. Using the two lists as samples from two populations, determine if the two samples are aproximately normal using appropriate hypothesis tests.
8. Using the two lists as samples from two populations, determine if the two samples appear to be from the same population or not using appropriate hypothesis tests.

4.1 Deliverables:

- Working scripts that produce plots
- Narrative (or print blocks) that supply answer questions
- CCMR citations for sources (Lab 14, Lesson 9, ... is OK for using internal sources, URL for outside sources)

4.2 Hints:

- A suggested set of code cells is listed below, the first cell is the roulette game, don't break it!
- Add/remove cells as needed for your solution

- You may find it useful to write a wrapper script to have the computer play with itself, but it is not required; it is ok if you play the 25 games manually, write down results, then hand enter them into a list

[15]:
```
#RUSSIAN ROULETTE PROGRAM IN PYTHON:

import random
print('THIS IS A RUSSIAN ROULETTE PROGRAM. BEST PLAYED WHILE DRINKING VODKA.')
leaveprogram=0
triggerpulls = 0
while leaveprogram != "q":
    print("Press Enter to Spin the Cylinder & Test Your Courage")
    input()
    number=random.randint (1, 6)
    if number==1:
        print("[ CLICK! ]")
        triggerpulls += 1
        print("Pulls = ",triggerpulls, "Type 'q' to quit")
        leaveprogram=input()
    if number==2:
        print("[ CLICK! ]")
        triggerpulls += 1
        print("Pulls = ",triggerpulls, "Type 'q' to quit")
        leaveprogram=input()
    if number==3:
        print("[ CLICK! ]")
        triggerpulls += 1
        print("Pulls = ",triggerpulls, "Type 'q' to quit")
        leaveprogram=input()
    if number==4:
        print("[ CLICK! ]")
        triggerpulls += 1
        print("Pulls = ",triggerpulls, "Type 'q' to quit")
        leaveprogram=input()
    if number==5:
        print("[ BANG!!!! ]")
        triggerpulls += 1
        print("[ So long ]")
        print("[ Comrade. ]")
        print("Pulls = ",triggerpulls)
        leaveprogram='q'
    if number==6:
        print("[ CLICK! ]")
        triggerpulls += 1
        print("Pulls = ",triggerpulls, "Type 'q' to quit")
        leaveprogram=input()

#
```

```
THIS IS A RUSSIAN ROULETTE PROGRAM. BEST PLAYED WHILE DRINKING VODKA.
Press Enter to Spin the Cylinder & Test Your Courage


[ CLICK! ]
Pulls =  1 Type 'q' to quit


Press Enter to Spin the Cylinder & Test Your Courage


[ CLICK! ]
Pulls =  2 Type 'q' to quit


Press Enter to Spin the Cylinder & Test Your Courage


[ CLICK! ]
Pulls =  3 Type 'q' to quit


Press Enter to Spin the Cylinder & Test Your Courage


[ CLICK! ]
Pulls =  4 Type 'q' to quit


Press Enter to Spin the Cylinder & Test Your Courage


[ CLICK! ]
Pulls =  5 Type 'q' to quit


Press Enter to Spin the Cylinder & Test Your Courage


[ CLICK! ]
Pulls =  6 Type 'q' to quit


Press Enter to Spin the Cylinder & Test Your Courage


[ CLICK! ]
Pulls =  7 Type 'q' to quit
```

```
Press Enter to Spin the Cylinder & Test Your Courage


[ BANG!!!! ]
[ So long ]
[ Comrade. ]
Pulls =  8
```

[16]:
```
# List of results
```

[17]:
```
# Histogram
```

[18]:
```
# Mean Pulls to Failure
```

[19]:
```
# Put Deer Hunter Version Here
```

[20]:
```
# List of results
```

[21]:
```
# Histogram
```

[22]:
```
# Mean Pulls to Failure
```

5 Problem 3 (15 points)

The data below are the impact impact strength of packaging materials in foot-pounds of two branded boxes. Produce a histogram of the two series, and determine if there is evidence of a difference in mean strength between the two brands. Use an appropriate hypothesis test to support your assertion at a level of significance of $\alpha = 0.10$.

Amazon Branded Boxes	Walmart Branded Boxes
1.25	0.89
1.16	1.01
1.33	0.97
1.15	0.95
1.23	0.94
1.20	1.02
1.32	0.98
1.28	1.06
1.21	0.98
1.14	0.94
1.17	1.02
1.34	0.98

5.1 Deliverables:

- Working scripts that produce perform the necessary tests
- Narrative (or print blocks) that supply answer questions
- CCMR citations for sources (Lab 14, Lesson 9, ... is OK for using internal sources, URL for outside sources)

5.2 Hints:

- A suggested set of code cells is listed below
- Add/remove cells as needed for your solution

[23]:
```
# define lists and make into dataframe
```

[24]:
```
# describe lists/dataframe
```

[25]:
```
# histograms
```

[26]:
```
# hypothesis tests are means same? (test for nomrality then t-test if normal;␣
↪mann-whitney if non-normal)
```

[27]:
```
# interpret findings (could be a markdown cell, or embed into code)
```

6 Problem 4 (35 points)

Precipitation records for Lubbock from 1895 to 2019 for the month of October is located at http://54.243.252.9/engr1330content/engr-1330-webroot/5-ExamProblems/Exam2/Exam2/spring2021/ the filename is `Lubbockdata.csv`

1. Produce a plot of year vs precipitation. *[Script + Plot 1: data==blue]*
2. Describe the entire data set. *[Script]*
3. Split the data into two parts at the year 1960. *[Script]*
4. Describe the two data series you have created. *[Script]*
5. Plot the two series on the same plot. *[Script + Plot 2: data1==blue, data2==green]*
6. Is there evidence of different mean precipitation in the pre-1990 and post-1990 data sets? Use a hypothesis test to support your assertion. *[Markdown + Script]*
7. Using the entire data set (before the 1960 split) prepare an empirical cumulative distribution plot using the weibull plotting position formula. *[Script + Plot 3: data==blue]*
8. What is the 50% precipitation exceedence depth? *[Markdown]*
9. What is the 90% precipitation exceedence depth? *[Markdown]*
10. Fit the empirical distribution using a normal distribution data model, plot the model using a red curve. Assess the fit. *[Script + Plot 4: data==blue, model==red]*
11. Fit the empirical distribution using a gammal distribution data model, plot the model using a red curve. Assess the fit. *[Script + Plot 5: data==blue, model==red]*
12. Using your preferred model (normal vs. gamma) estimate the 99% precipitation exceedence depth. *[Script + Markdown]*

6.1 Deliverables:

- Working scripts that perform the necessary steps listed in the 12 steps above
- Narrative (or print blocks) that supply answer questions
- CCMR citations for sources (Lab 14, Lesson 9, ... is OK for using internal sources, URL for outside sources)

6.2 Hints:

- The 12 steps above suggest content and cell type; script cells are for code, Markdown for narrative
- If you wish to use print statements instead of markdown that's fine!
- A suggested set of code cells is listed below
- Add/remove cells as needed for your solution

[28]:
```
# read the file
```

[29]:
```
# display a few lines
```

[30]:
```
# plot data
```

[31]:
```
# describe data
```

[32]:
```
# split into two series
```

[33]:
```
# describe each of the split series
```

[34]:
```
# plot the split series (blue == old) (green == new)
```

[35]:
```
# compare sample means
```

[36]:
```
# plotting position function OK to copy from lab
```

[37]:
```
# Make a data list for plotting position function
```

[38]:
```
# Generate quantile values from pp function
```

[39]:
```
# Make a quantile plot OK to copy from lab and modify for this problem
```

[40]:
```
# Visual interpretation (in a Markdown cell )
# 50% is at about ??? inch depth
# 90% is at about ??? inch depth
```

[41]:
```
# normal distribution data model copy from lesson/lab
```

[42]:
```
# Fitted Model copy from lesson/lab
```

[43]:
```
# Now plot the sample values and plotting position
# Built the plot
```

[44]:
```
# gamma distribution data model
```

[45]:
```
# Fitted Model copy from lesson/lab
```

[46]:
```
# Now plot the sample values and plotting position
# Built the plot
```

[47]:
```
# Choose best model
# Capture parameters
# Estimate value at quantile = 0.99
```

---

7 Bonus Problem (5 pts)

Consider the script below, which implements a simulation of Russian Roulette (using an object oriented approach). Run the script to familarize yourself with the output.

Then to prevent `Farhang` from dying, determine a way to change his outcome, and explain how you save him.

Like with the robot speeding ticket you are channeling Kirk's approach to the Kobayashi-Maru exercise https://en.wikipedia.org/wiki/Kobayashi_Maru You can also find the necessary trick from https://en.wikipedia.org/wiki/WarGames

[57]:
```
import random
import itertools

class RussianRoulette:

    def __init__(self, players, chambers=6):
        random.shuffle(players)
        self.players = itertools.cycle(players)
        self.chambers = [False for _ in range(chambers)]
        self.current = None
        self.rounds = 0

    def load(self):
        """
        Randomly load a chamber with a bullet.
        """
        chamber_to_load = random.randint(0, len(self.chambers)-1)

        for i, chamber in enumerate(self.chambers):
```

```
            if i == chamber_to_load:
              self.chambers[i] = True

    def next_round(self):
        """
        Advance to the next round.

        Returns:
            the `player` whose turn it is next
        """
        self.rounds += 1
        return next(self.players)

    def spin(self):
        """
        Randomly assign a new chamber.
        """
        self.current = random.randrange(0, len(self.chambers))

    def fire(self, player):
        """
        Fires the gun, then advances to the next chamber.

        The gun will loop back to the first chamber if we were at the
        final chamber in the cyclinder.

        Returns:
            `None` if no one has died
            `player` if the bullet was in the next chamber
        """
        if self.chambers[self.current]:
            return player

        self.current = (self.current + 1) % len(self.chambers)

if __name__ == '__main__':
    players = ['Nikita', 'Dima', 'Sergey', 'Farhang', 'Andrey',  'Neko',␣
 ↪'Cleveland']
    game = RussianRoulette(players)

    game.load()
    game.spin()

    while True:
        player = game.next_round()

        choice = random.choice(['spin', 'fire'])
```

```
        if choice == 'spin':
            game.spin()
            #pass

        if game.fire(player):
            print(f'{player} died. :(')
            print(f'{game.rounds} completed.')
            break

        print(f'{player} lives to see another round!')
```

```
Farhang lives to see another round!
Andrey lives to see another round!
Nikita lives to see another round!
Dima lives to see another round!
Sergey lives to see another round!
Neko lives to see another round!
Cleveland lives to see another round!
Farhang lives to see another round!
Andrey lives to see another round!
Nikita lives to see another round!
Dima lives to see another round!
Sergey died. :(
12 completed.
```

---

8 Bonus Problem (10 points):

1. Write a function to find the probability of an event in percentage form based on given outcomes and sample space
    1. Use the function and compute the probability of rolling a number between 12 and 18 (inclusive) with a D20
    2. Use the function and compute the probability of drawing a King of hearts after drawing a card of diamonds from a standard deck of cards (without replacement)
    3. Use the function and compute the probability of drawing a Queen or a King or a 5 of spades.

---

9 Bonus Problem (30 points)

Write a script for the Treasure Hunt problem (repeated below).

Tresure Hunt Problem

34	21	32	41	25
14	42	43	14	31
54	45	52	42	23
33	15	51	31	35
21	52	33	13	23

Do you like treasure hunts? In this problem you are to write a program to explore the above array for a treasure. The values in the array are clues. Each cell contains an integer between 11 and 55; for each value the ten's digit represents the row number and the unit's digit represents the column number of the cell containing the next clue. Starting in the upper left corner (at 1,1), use the clues to guide your search of the array. (The first three clues are 34, 42, 15). The treasure is a cell whose value is the same as its coordinates. Your program must first read in the treasure map data into a 5 by 5 array. Your program should output the cells it visits during its search, and a message indicating where you found the treasure.

The "Treasure Hunt Problem" is from the HackerRank.com avaiable at https://www.hackerrank.com/contests/startatastartup/challenges/treasure-hunt

Two treasure maps are located at http://54.243.252.9/engr1330content/engr-1330-webroot/5-ExamProblems/Exam2/Exam2/spring2021/ the filenames are `treasure1.txt` and `treasure2.txt` Test your program using both maps and capture the output.

[49]:
```
# read the map
```

[50]:
```
# parse the map
```

[51]:
```
# start at pos[0,0] and follow map directions until find treasure, print output␣
↪along the search
```

[58]:
```

```

[ ]:
```

```