# Lecture 27 – Final Review

## DSC 10, Spring 2023

# Announcements

- The Final Exam is **this Saturday from 7-10PM**. Read more details here.
    - Study by working through old exams at practice.dsc10.com as if they are real exams – give yourself only 3 hours, and only refer to the reference sheet.
- If at least 80% of the class fills out both CAPEs and the End-of-Quarter Survey, then the entire class gets 0.5% of extra credit on their overall grade. We value your feedback!

DEADLINE:
SATURDAY 8AM!

Currently below 50%!

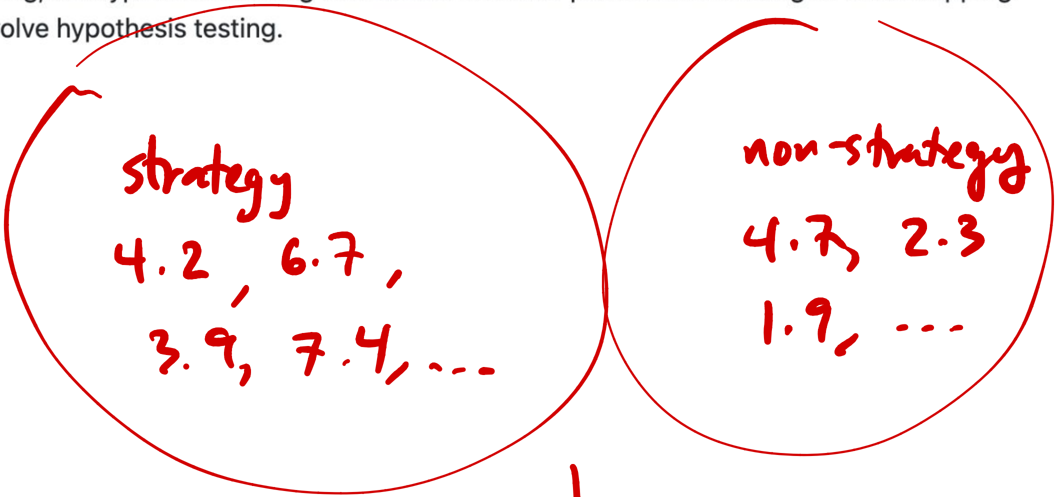
# Agenda

- The goal is cover Problems 6 through 14 from the Winter 2023 Final Exam.
    - We'll cover half in the 12-12:50PM section, half in the 1-1:50PM section.
    - If we finish early, we'll cover other problems on the exam.
- Open the exam on your device and write down the solutions along with me.
- **Start by reading the data info sheet!**

# Problem 6

Choose the best tool to answer each of the following questions. Note the following:

- By “hypothesis testing,” we mean “standard” hypothesis testing, i.e. hypothesis testing that **doesn’t** involve permutation testing or bootstrapping.
- By “bootstrapping,” we mean bootstrapping that **doesn’t** involve hypothesis testing.



## Problem 6.1

Are strategy games rated higher than non-strategy games?

○ Hypothesis testing

[✓ Permutation testing]

○ Bootstrapping

yes/no question

↳

| category | rating |
|---|---|
| strategy | 4.2 |
| non-strat | 4.7 |
| non-strat | 2.3 |
| strategy | 6.7 |
| ⋮ | ⋮ |

## Problem 6.2

What is the mean complexity of all games?

○ Hypothesis testing

○ Permutation testing

[✓ Bootstrapping]

estimating a pop. parameter

# Problem 6.3

Are there an equal number of cooperative and non-cooperative games?

[✓ Hypothesis testing]

○ Permutation testing

○ Bootstrapping

43 coop

71 non-coop

→ Does this sample look like it came from [50%, 50%]?

→ kind of like coin flipping.

# Problem 6.4

Are games with more than one domain more complex than games with one domain?

○ Hypothesis testing

[✓ Permutation testing]

○ Bootstrapping

| domains | complexity |
|---|---|
| one | 2.4 |
| more | 4.7 |
| more | ⋮ |
| one | ⋮ |
| one | |

# Problem 7

We use the regression line to predict a game's "Rating" based on its "Complexity". We find that for the game *Wingspan*, which has a "Complexity" that is 2 points higher than the average, the predicted "Rating" is 3 points higher than the average.

$\uparrow y$ $\uparrow x$

Regression line SUs

$$\text{pred } y_{(su)} = r \cdot x_{(su)}$$

$$\frac{\text{pred } y - \text{mean}(y)}{SD(y)} = r \cdot \frac{x - \text{mean}(x)}{SD(x)}$$

$$\frac{3}{SD(y)} = r \cdot \frac{2}{SD(x)}$$

## Problem 7.1

What can you conclude about the correlation coefficient r?

○ $r < 0$

○ $r = 0$

[✓ $r > 0$]

○ We cannot make any conclusions about the value of $r$ based on this information alone.

## Problem 7.2

What can you conclude about the standard deviations of “Complexity” and “Rating”?

[✓ SD of "Complexity" < SD of "Rating"]

○ SD of "Complexity" = SD of "Rating"

○ SD of "Complexity" > SD of "Rating"

○ We cannot make any conclusions about the relationship between these two standard deviations based on this information alone.

$$3 \cdot SD(x) = 2r\, SD(y)$$

$$SD(x) = \frac{2r}{3}\, SD(y)$$

between 0 and 2/3

$$\Rightarrow SD(x) < SD(y)$$

# Problem 8

$x$ $y$ $-1 < r < 0$

Suppose that for children's games, `"Play Time"` and `"Rating"` are negatively linearly associated due to children having short attention spans. Suppose that for children's games, the standard deviation of `"Play Time"` is twice the standard deviation of `"Rating"`, and the average `"Play Time"` is 10 minutes. We use linear regression to predict the `"Rating"` of a children's game based on its `"Play Time"`. The regression line predicts that *Don't Break the Ice*, a children's game with a `"Play Time"` of 8 minutes will have a `"Rating"` of 4. Which of the following could be the average `"Rating"` for children's games?

regression line original units

$$y = mx + b \qquad m = r \cdot \frac{SD(y)}{SD(x)} = \frac{r}{2}$$

$$SD(x) = 2\,SD(y) \Rightarrow \frac{SD(y)}{SD(x)} = \frac{1}{2}$$

$$b = \text{mean}(y) - m \cdot \text{mean}(x)$$

$$= \text{mean}(y) - \frac{r}{2} \cdot 10 = \text{mean}(y) - 5r$$

○ 2

○ 2.8

[✓ 3.1]

○ 4

$$y = mx + b$$

$$4 = 8m + b$$

$$4 = 8 \cdot \frac{r}{2} + \text{mean}(y) - 5r$$

$$4 = 4r - 5r + \text{mean}(y) \Rightarrow 4 + r = \text{mean}(y)$$

$$-1 + 4 < r + 4 < 4 \rightarrow 3 < \text{mean}(y) < 4$$

# Problem 9

The function `perm_test` should take three inputs:

1. `df`, a DataFrame.
2. `labels`, a string. The name of a column in df that contains two distinct values, which signify the groups in a permutation test.
3. `data`, a string. The name of a column in df that contains numerical data.

The function should return an array of 1000 simulated differences of group means, under the assumption of the null hypothesis in a permutation test, namely that data in both groups come from the same population.

The smaller of the two group labels should be first in the subtraction. For example, if the two values in the `labels` column are `"dice game"` and `"card game"`, we would compute the difference as the mean of the `"card game"` group minus the mean of the `"dice game"` group, because `"card game"` comes before `"dice game"` alphabetically. Note that **`groupby`** orders the rows in ascending order by default.

An incorrect implementation of `perm_test` is provided below.

```
def perm_test(df, labels, data):
  diffs = np.array([])
  for i in np.arange(1000):
      df.assign(shuffled=np.random.permutation(df.get(data)))
      means = df.groupby(labels).mean().get(data)
      diff = means.iloc[0] - means.iloc[1]
      diffs = np.append(diffs, diff)
      return diffs
```

① df = needs to come in front

② .get("shuffled") instead of .get(data)

③ un-indent the return

Three lines of code above are incorrect. Your job is to identify which lines of code are incorrect, and describe briefly in English how you would fix them. You don't need to explain why the current code is wrong, just how to fix it.

## Problem 9.4

Write one line of code that creates an array called `simulated_diffs` containing 1000 simulated differences in group means for this permutation test. You should call your `perm_test` function here!

perm_test(with_dice[with_dice.get("Domains").str.contains("Children's Games")], "isDice", "Play Time")

## Problem 9.5

Suppose we've stored the observed value of the test statistic for this permutation test in the variable `obs_diff`. Fill in the blank below to find the p-value for this permutation test.

np.count_nonzero(simulated_diffs ___ obs_diff) / 1000

~~○ <~~

~~○ <=~~

○ >

[✓ ○ >=]

→ test stat = False mean − True mean

alt : True mean < False mean

→ ∴ we want test stat to be big!

# Problem 10

population

It's your first time playing a new game called *Brunch Menu*. The deck contains 96 cards, and each player will be dealt a hand of 9 cards. The goal of the game is to avoid having certain cards, called *Rotten Egg* cards, which come with a penalty at the end of the game. But you're not sure how many of the 96 cards in the game are *Rotten Egg* cards. So you decide to use the Central Limit Theorem to estimate the proportion of Rotten Egg cards in the deck based on the 9 random cards you are dealt in your hand.

↓ sample

$[\underbrace{\quad}_{-2SD} \bullet \underbrace{\quad}_{+2SD}]$

95% CI: sample mean ± $2 \cdot \underbrace{\text{SD of sample mean's distribution}}$

## Problem 10.1

You are dealt 3 Rotten Egg cards in your hand of 9 cards. You then construct a CLT-based 95% confidence interval for the proportion of Rotten Egg cards in the deck based on this sample. Approximately, how wide is your confidence interval?

$$\frac{\text{Pop SD}}{\sqrt{\text{size}}} \approx \frac{\text{Sample SD}}{\sqrt{\text{size}}}$$

Choose the closest answer, and use the following facts:

- The standard deviation of a collection of 0s and 1s is $\sqrt{(\text{Prop. of 0s}) \cdot (\text{Prop of 1s})}$.
- $\sqrt{18}$ is about $\frac{17}{4}$.

$$\text{width} = 4 \cdot \frac{\text{Sample SD}}{\sqrt{\text{size}}} = 4 \cdot \frac{\sqrt{\frac{6}{9} \cdot \frac{3}{9}}}{\sqrt{9}} = 4 \cdot \frac{\frac{\sqrt{18}}{9}}{3}$$

$$= \frac{4\sqrt{18}}{27} \approx \frac{\cancel{4} \cdot \frac{17}{\cancel{4}}}{27} = \frac{17}{27}$$

○ $\frac{17}{9}$

[✓ $\frac{17}{27}$]

○ $\frac{17}{81}$

○ $\frac{17}{96}$

# Problem 10.2

Which of the following are limitations of trying to use the Central Limit Theorem for this particular application? Select all that apply.

correct

[✓ □] The CLT is for large random samples, and our sample was not very large.

correct

[✓ □] The CLT is for random samples drawn with replacement, and our sample was drawn without replacement.

☒ ~~The CLT is for normally distributed data, and our data may not have been normally distributed.~~

wrong: CLT works for any population

☒ ~~The CLT is for sample means and sums, not sample proporti~~ons.

wrong: proportions are means!

```
1  def wham(a, b):
2      if a < b:
3          return a + 2
4      if a + 2 == b:
5          print(a + 3)
6          return b + 1
7      elif a - 1 > b:
8          print(a)
9          return a + 2
10     else:
11         return a + 1
```

## Problem 11.2

Give an example of a pair of integers `a` and `b` such that `wham(a, b)` returns `a + 1`.

wham $\boxed{(5,4)}$

$\underbrace{b \le a < b+1}$

## Problem 11.3

Which of the following lines of code will never be executed, for any input?

○ 3

[✓ 6]

○ 9

○ 11

if a+2==b, then a<b, and we already returned!

## Problem 11.1

What is printed when we run `print(wham(6, 4))`?

a b

6
8

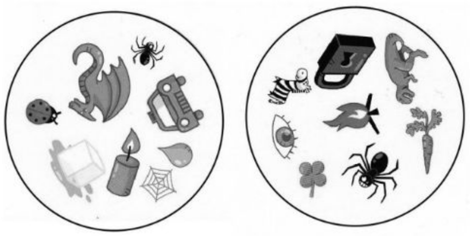
In [3]: wham(6,4)

6

Out[3]: 8

# Problem 12

In the game *Spot It*, players race to identify an object that appears on two different cards. Each card contains images of eight objects, and exactly one object is common to both cards.



## Problem 12.1

Suppose the objects appearing on each card are stored in an array, and our task is to find the object that appears on both cards. Complete the function `find_match` that takes as input two arrays of 8 objects each, with one object in common, and returns the name of the object in both arrays.

For example, suppose we have two arrays defined as follows.

```
objects1 = np.array(["dragon", "spider", "car", "water droplet", "spiderweb", "candle", "ice c
objects2 = np.array(["zebra", "lock", "dinosaur", "eye", "fire", "shamrock", "spider", "carrot
```

Then `find_match(objects1, objects2)` should evaluate to `"spider"`. Your function must include a for loop, and it must take **at most three lines of code** (not counting the line with `def`).

# Problem 12.2

Now suppose the objects appearing on each card are stored in a DataFrame with 8 rows and one column called `"object"`. Complete the function `find_match_again` that takes as input two such DataFrames with one object in common and returns the nameof the object in both DataFrames.

Your function may not call the previous function `find_match`, and it must take exactly **one line of code** (not counting the line with `def`).

# Problem 13

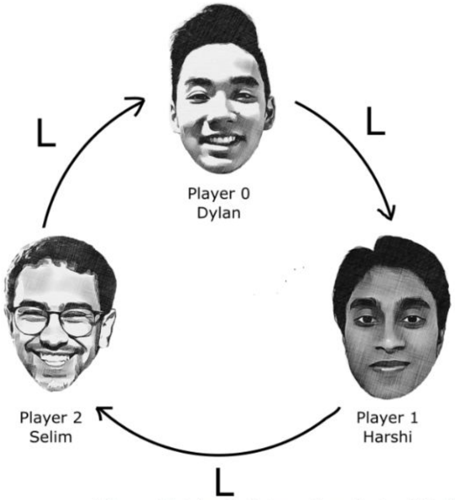
Dylan, Harshi, and Selim are playing a variant of a dice game called *Left, Center, Right* (*LCR*) in which there are 9 chips (tokens) and 9 dice. Each player starts off with 3 chips. Each die has the following six sides: L, C, R, Dot, Dot, Dot.

During a given player's turn, they must roll a number of dice equal to the number of chips they currently have. Each die determines what to do with one chip:

1. L means give the chip to the player on their left.
2. R means give the chip to the player on their right.
3. C means put the chip in the center of the table. This chip is now out of play.
4. Dot means do nothing (or keep the chip).

Since the number of dice rolled is the same as the number of chips the player has, the dice rolls determine exactly what to do with each chip. There is no strategy at all in this simple game.

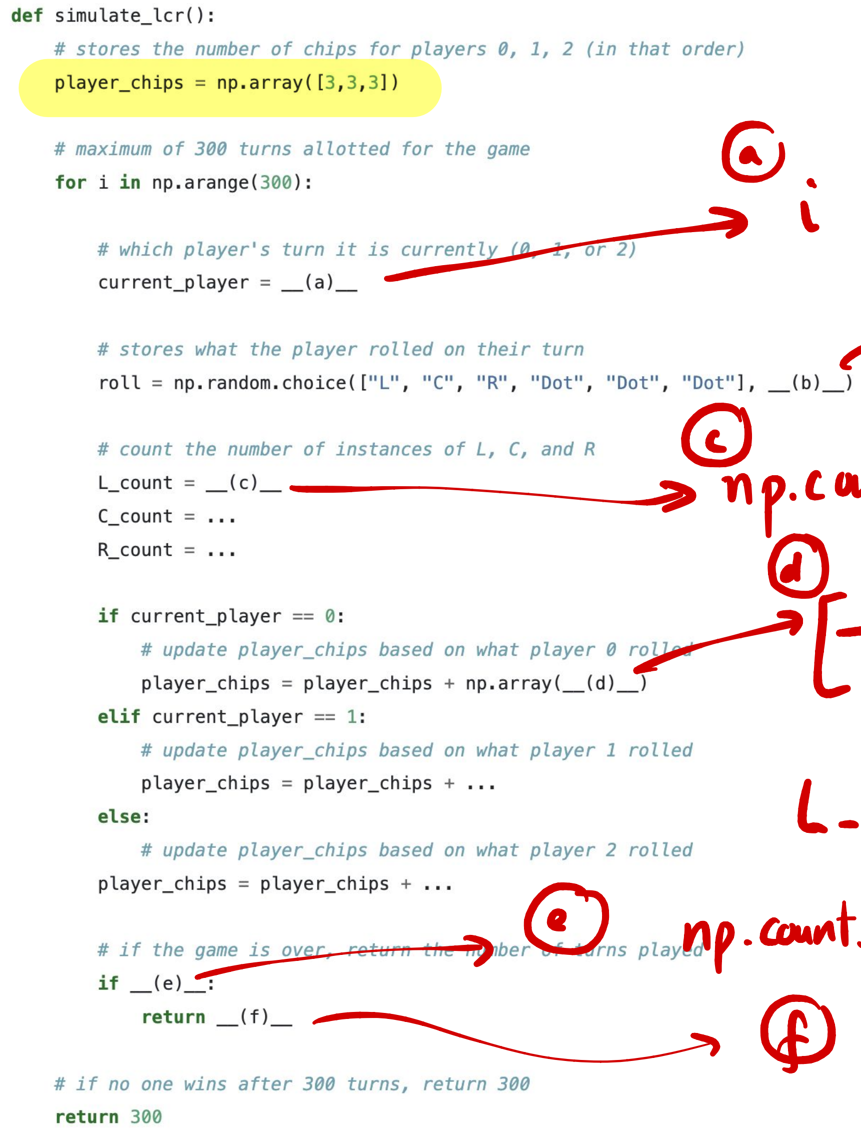
Dylan will take his turn first (we'll call him Player 0), then at the end of his turn, he'll pass the dice to his left and play will continue clockwise around the table. Harshi (Player 1) will go next, then Selim (Player 2), then back to Dylan, and so on.



Note that if someone has no chips when it's their turn, they are still in the game and they still take their turn, they just roll 0 dice because they have 0 chips. The game ends when only one person is holding chips, and that person is the winner. If 300 turns have been taken (100 turns each), the game will end and we'll declare it a tie.

The function `simulate_lcr` below simulates one full game of *Left, Center, Right* and returns the number of turns taken in that game. Some parts of the code are not provided. You will need to fill in the code for the parts marked with a blank. The parts marked with `...` are not provided, but you don't need to fill them in because they are very similar to other parts that you do need to complete.

**Hint:** Recall that in Python, the % operator gives the remainder upon division. For example 12 % 5 is 2.



```
def simulate_lcr():
    # stores the number of chips for players 0, 1, 2 (in that order)
    player_chips = np.array([3,3,3])

    # maximum of 300 turns allotted for the game
    for i in np.arange(300):

        # which player's turn it is currently (0, 1, or 2)
        current_player = __(a)__

        # stores what the player rolled on their turn
        roll = np.random.choice(["L", "C", "R", "Dot", "Dot", "Dot"], __(b)__)

        # count the number of instances of L, C, and R
        L_count = __(c)__
        C_count = ...
        R_count = ...

        if current_player == 0:
            # update player_chips based on what player 0 rolled
            player_chips = player_chips + np.array(__(d)__)
        elif current_player == 1:
            # update player_chips based on what player 1 rolled
            player_chips = player_chips + ...
        else:
            # update player_chips based on what player 2 rolled
        player_chips = player_chips + ...

        # if the game is over, return the number of turns played
        if __(e)__:
            return __(f)__

    # if no one wins after 300 turns, return 300
    return 300
```

(a) i % 3

(b) player_chips[current_player]

(c) np.count_nonzero(roll == "L")

you lose the L, C, and R chips

(d) [-L_count
- C_count
- R_count
L_count, R_count]

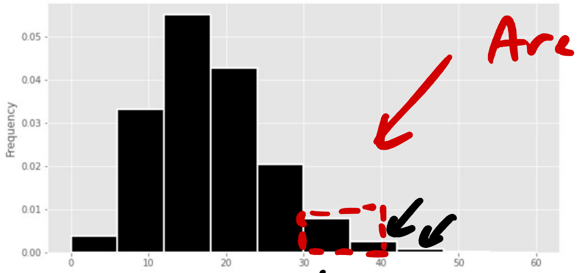
[3, 3, 3]
0 1 2

(e) np.count_nonzero(player_chips) == 1

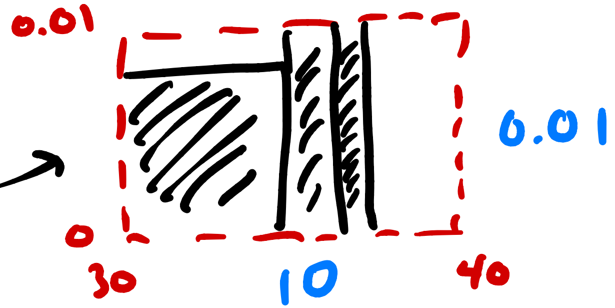
(f) i+1

Consider the code and histogram below.

```
turns = np.array([])
for i in np.arange(10000):
    turns = np.append(turns, simulate_lcr())
(bpd.DataFrame().assign(turns=turns).plot(kind="hist", density = True, ec="w", bins = np.arange(0, 66, 6)))
```



Area to the right of 30



## Problem 14.1

Does this histogram show a probability distribution or an empirical distribution?

○ Probability Distribution

[✓ Empirical Distribution] → simulations result in empirical distributions!

## Problem 14.2

What is the probability of a game of *Left, Center, Right* lasting 30 turns or more? Choose the closest answer below.

○ ~~0.01~~

[✓ 0.06]

○ 0.10

○ ~~0.60~~

## Problem 14.3

Suppose a player with $n$ chips takes their turn. What is the probability that they will have to put at least one chip into the center? Give your answer as a mathematical expression involving $n$.

P($\overset{\text{single}}{\text{chip}}$ goes to center) $= \frac{1}{6}$

P(at least one chip goes to center) $= 1 - $ P(none go to center)

$$= 1 - \left(1-\frac{1}{6}\right) \times \left(1-\frac{1}{6}\right) \times \cdots$$

$$= 1 - \left(\frac{5}{6}\right)^n$$

## Problem 14.4

Suppose a player with $n$ chips takes their turn. What is the probability that they will end their turn with $n$ chips? Give your answer as a mathematical expression involving $n$.

P(keep single chip) $= \frac{3}{6} = \frac{1}{2}$

P(keep all chips) $= \frac{1}{2} \cdot \frac{1}{2} \underbrace{\cdot \ldots \cdot}_{n \text{ times}} \frac{1}{2} = \left(\frac{1}{2}\right)^n$

## Problem 16

We collect data on the play times of 100 games of *Chutes and Ladders* (sometimes known as *Snakes and Ladders*) and want to use this data to perform a hypothesis test.

## Problem 16.1

Which of the following pairs of hypotheses can we test using this data?

Option 1: **Null Hypothesis:** In a random sample of Chutes and Ladders games, the average play time is 30 minutes. **Alternative Hypothesis:** In a random sample of Chutes and Ladders games, the average play time is not 30 minutes.

Option 2: **Null Hypothesis:** In a random sample of Chutes and Ladders games, the average play time is not 30 minutes. **Alternative Hypothesis:** In a random sample of Chutes and Ladders games, the average play time is 30 minutes

Option 3: **Null Hypothesis:** A game of Chutes and Ladders takes, on average, 30 minutes to play. **Alternative Hypothesis:** A game of Chutes and Ladders does not take, on average, 30 minutes to play.

Option 4: **Null Hypothesis:** A game of Chutes and Ladders does not take, on average, 30 minutes to play. **Alternative Hypothesis:** A game of Chutes and Ladders takes, on average, 30 minutes to play.

- ○ Option 1
- ○ Option 2
- ○ Option 3
- ○ Option 4

## Problem 16.2

We use our collected data to construct a 95% CLT-based confidence interval for the average play time of a game of *Chutes and Ladders*. This 95% confidence interval is [26.47, 28.47]. For the 100 games for which we collected data, what is the mean and standard deviation of the play times?

## Problem 16.3

Does the CLT say that the distribution of play times of the 100 games is roughly normal?

○ Yes

○ No

## Problem 16.4

Of the two hypotheses you selected in part (a), which one is better supported by the data?

○ Null Hypothesis

○ Alternative Hypothesis