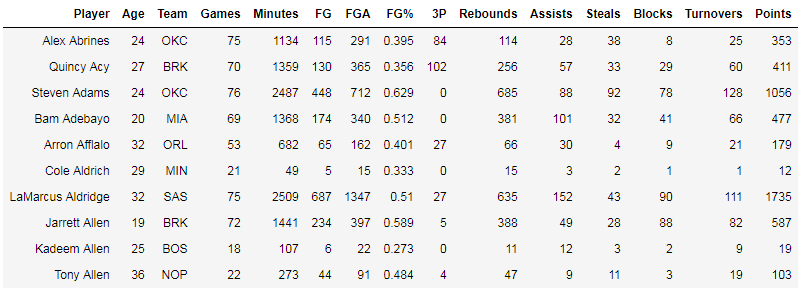
Week 8: Midterm Review

Data 8 Tutoring

# Midterm Review

**1. Table Practice**

The nba table contains data from the 2017-2018 season for every active player. Each row represents statistics over the whole season. In this table, percentages are expressed as decimals.



1. eFG% is an advanced statistic commonly used over FG% (field goal percentage). Calculate eFG% using the formula (FG + 0.5\*3P)/FGA. Make sure to use FG and not FG%.

Once it’s been calculated, append the values as the column eFG% to this table.

numerator = nba.column(“FG”) + 0.5\*nba.column(“3P”)

denominator = nba.column(“FGA”)

nba\_efg = nba.with\_column(“eFG%”, numerator / denominator)

2. Find the team with the lowest average eFG% (return the name only)

by\_team = nba\_efg.group(“Team”, np.mean)

by\_team.sort(“eFG% mean”).column(“Team”).item(0)

3. What proportion of points scored were by players who had an eFG% above 60%? The variable answer should be your final proportion.

more\_than\_sixty = sum(nba\_efg.where(“eFG%”,

are.above(0.6)).column(“Points”))

total = sum(nba\_efg.column(“Points”))

answer = more\_than\_sixty / total

**2. Coding Practice**

1. Suppose a broken candy machine dispenses sweet candy 99% of the time and sour candy otherwise. What is the chance you find at least 1 sour candy in 50 candies dispensed randomly from the machine? Let candies be an array with the first element containing the probability of picking a sweet candy and the second element being the probability of picking sour candy. Use simulation to estimate the probability.

candies = make\_array(0.99, 0.01)

sour = \_\_\_\_\_\_\_\_\_\_\_

for i in \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_(5000):

chosen\_candies\_prop = sample\_proportions(\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_)

sour\_prop = \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

if \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_:

sour = sour + 1

chance\_of\_at\_least\_one = \_\_\_\_\_\_\_\_\_\_/5000

candies = make\_array(0.99, 0.01)

sour = 0

for i in np.arange(5000):

chosen\_candies\_prop = sample\_proportions(50, candies)

sour\_prop = chosen\_candies\_prop.item(1)

if sour\_prop > 0:

sour = sour + 1

chance\_of\_at\_least\_one = sour/5000

2. What is the exact probability of finding at least 1 sour candy in 50 candies dispensed from the broken candy machine?

Hint: think about the complement rule!

By the complement rule, we know that:

P(finding at least 1 sour candy in 50 candies) = 1 - P(finding NO sour candies).

Therefore, we can just calculate P(finding NO sour candies), which equals (0.99)^50, since we need to get a sweet candy on the first draw, a sweet candy on the second draw, etc for 50 draws. This means

P(finding at least 1 sour candy in 50 candies) = 1 - (0.99)^50

**3. Defining Functions**

Suppose you have a table dinners, which contains a row for every dinner eaten at a given restaurant for a week. dinners has a column “Day” of strings corresponding to which day of the week the dinner was eaten, and a column “Subtotal” of integers containing costs without tips.

1. Define a function called compute\_tip that given a total bill as an integer, returns a 20% tip calculated from the bill.

def compute\_tip(bill):

return \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

def compute\_tip(bill):

return bill \* 0.2

2. Add a new column to the dinners table called “Tip” that is the 20% tip for each bill in the “Subtotal” column and name the resulting table dinners\_with\_tip.

dinners\_with\_tip = dinners.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_(“Tip”,

dinners.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_(\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_, \_\_\_\_\_\_\_\_\_\_\_\_))

dinners\_with\_tip = dinners.with\_column(“Tip”,

dinners.apply(compute\_tip, “Subtotal”))

3. Write code to calculate the day of the week with the lowest average cost (not including tip).

by\_day = dinners.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_(\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_, \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_)

by\_day.sort(\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_).\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_(\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_).item(0)

by\_day = dinners.group(“Day”, np.mean)

by\_day.sort(“Subtotal mean”).column(“Day”).item(0)

**4. Probability and Hypothesis Testing**

Your good friend Gary is playing with a fair 6-sided die.

1. Gary rolls the die 10 times and rolls a 6 every time. What is the probability of that event occurring?

P(rolling ten 6’s in ten rolls) = (⅙)^10 (Product Rule)

1. Then, Gary rolls the die twice. What is the probability he rolls a 1 on the first roll and a 2 on the second roll?

(⅙)\*(⅙) because a “1” face and “2” each have a ⅙ chance of being rolled, and since we want both of these faces to occur, we must multiply the probabilities.

1. Suppose Gary rolls the die 5 times - what is the probability he rolls at least one 3?

P(at least one 3 in 5 rolls) = 1 - P(no 3s in 5 rolls) = 1 - (⅚)^5

1. If you are considering whether the die is fair, do the results of the throws represent numerical or categorical outcomes?

Categorical outcomes, since we don’t actually care about the number that we are rolling (like rolling a two or a four); rather, we want to compare the count of times each face is rolled. Therefore, the results of throws could be treated as categorical outcomes.

**5. Hypothesis Testing**

Gary rolls the die another 10 times and rolls 7 sixes, 2 fours, and 1 two. I suspect Gary is using an unfair die and I want to do a hypothesis test to check this.

a) Specify a null and alternative hypothesis.

Null Hypothesis: The die is fair and each roll follows the distribution of (⅙, ⅙, ⅙, ⅙,⅙, ⅙ ). Any variation is due to random chance.

Alternative Hypothesis: The die is not fair.

b) What test statistic would you choose to compare the null distribution above with what you simulate repeatedly to test whether the die is fair? Explain.

We could use TVD, since we are comparing the difference between **two categorical distributions**. In this case, we could compare the simulated distribution (proportions of each face we roll) vs the expected distribution if the die was fair (answer from above).

c) Calculate the obs\_test\_stat.

fair\_die = make\_array(1/6, 1/6, 1/6, 1/6, 1/6, 1/6)

gary\_die = make\_array(0, 1/10, 0, 2/10, 0, 7/10)

obs\_test\_stat = 0.5\*sum(abs(fair\_die-gary\_die))

obs\_test\_stat

d) Fill in the code below to simulate the distribution of faces if we roll a fair die 10 times and calculate one test statistic.

def calculate\_test\_stat():

fair\_die = make\_array(1/6, 1/6, 1/6, 1/6, 1/6, 1/6)

simulated\_die = sample\_proportions(10, fair\_die)

test\_stat = 0.5\*sum(abs(fair\_die-simulated\_die))

return test\_stat

e) Fill in the code below to simulate 10000 test statistics and generate the following histogram.

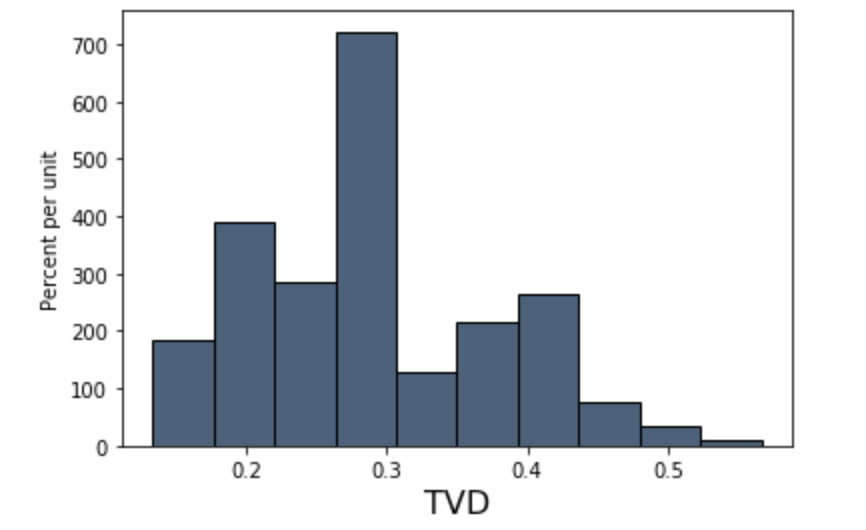
test\_stats = make\_array()

for i in np.arange(1000):

sim\_test\_stat = calculate\_test\_stat()

test\_stats = np.append(test\_stats, sim\_test\_stat)

Table().with\_column('TVD', test\_stats).hist(‘TVD’)



f) Fill in the code below to calculate the p-value.

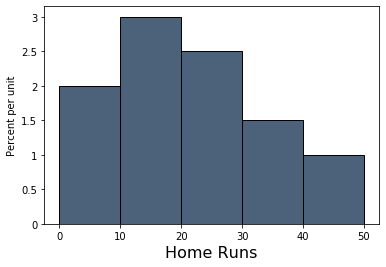
p\_value = np.count\_nonzero(test\_stats>=obs\_test\_stat)/len(test\_stats)

g) If p\_value turns out to be 0.004 and we use a p-value cutoff of 0.05, what conclusion do we draw from this test?

Since the p\_value is less than the p\_value cutoff, our data is more consistent with the alternative hypothesis and we reject the null hypothesis in favor of the alternative hypothesis.

**6. Histograms**

The following is a histogram of the number of home runs hit by MLB players in the 2019 season.



Answer the following questions using the histogram above. If it is not possible to compute the answer, write “Not Possible” and explain why you cannot calculate the answer.

a) Find the percent of players who hit between 10 and 20 (not inclusive) home runs in 2019.

(20 - 10) \* 3% = 30%

b) Find the number of players who hit more than 30 home runs in 2019.

Not possible since we do not know the number of players.

c) What percent of players hit at least 20 home runs?

100% - (10 - 0) \* 2% - (20 - 10) \* 3% = 100% - 50% = 50%

d) How many players hit between 25 and 30 home runs?

Not possible since we do not know the distribution of home runs within a bin.

e) We decide that we want more information about the players that hit between 10 and 20 home runs, so we split the [10,20) bin into two bins: a [10, 15) bin and a [15, 20) bin. We find that 20% of players hit between 10 and 15 home runs. What is the height of this new [15, 20) bin?

We know that the total area of the [10, 20) bin is 30% (found in part a). This means that the area of the [15, 20) bin is 30% - 20% = 10%. Using the area = height \* bin width formula:

10% = height \* 5 home runs

10% / 5 home runs = height → height = 2% / home run