UNIVERSITY OF TORONTO
Faculty of Arts & Science

STA130 F23 Midterm Examination

Friday, Nov 3, 2023

Duration: 1 hour and 50 minutes starting 10 minutes after the hour.
Aids Allowed: One "normal" $11 \times 8\frac{1}{2}$ page "cheatsheet", any size, any font, front and back.
Instructions: Write answers in the space provided in the exam and mark multiple choice answers on the scantron sheet at the end of the exam to match the exam question numbers.

Exam Reminders:

- Fill out your name and UTORid on this page and on the scantron answer sheet attached as the last page of the exam.
- Do not begin writing the actual exam until the announcements have ended and the Exam Facilitator has started the exam.
- As a student, you help create a fair and inclusive writing environment. If you possess an unauthorized aid during an exam, you may be charged with an academic offence.
- Turn off and place all cell phones, smart watches, electronic devices, and unauthorized study materials in your bag under your desk. If it is left in your pocket, it may be an academic offence.
- When you are done your exam, raise your hand for someone to come and collect your exam. Do not collect your bag and jacket before your exam is handed in.
- If you are feeling ill and unable to finish your exam, please bring it to the attention of an Exam Facilitator so it can be recorded before leaving the exam hall.
- In the event of a fire alarm, do not check your cell phone when escorted outside.

0. To keep things fair, no questions about the exam will be answered during the duration of the exam. If you think there is a problem with a question, note the question and briefly describe the problem in the space below and your concern will be evaluated during marking.

1. Which of the following tests whether or not two numbers *are equal* in `python`? B

   A. `a = b` B. `a == b` C. `a != b` D. `a >= b`

2. Which of the following imports `pandas` into `python` using its standard *alias*? E

   A. `import pandas` B. `import pandas as np`
   C. `import pandas pd` D. `import pandas as alias` E. `import pandas as pd`

3. What is the data type of the `python` object `(1338, 36)`? E

   A. `dict` B. `float` C. `int` D. `list` E. `tuple`

4. What number would replace the question mark in `(1338, 36)[?]` in order to access the value `36`?

   A. `0` B. `1` C. `2` D. `36` B

5. In `python` the expression `True + False` evaluates to `1`. What is this behavior called? B

   A. Boolean selection B. Coercion C. Conditional Logic

   D. None of the above because `python` is zero-indexed so this evaluates to `0`

6. Which of the following is the `pandas` expression which counts the number of missing (`nan`) values in `df` a `DataFrame` object? E

   A. `df.isnull()` B. `df.isnull().count()`
   C. `df.describe().isnull()` D. `(df is np.NaN).count()` E. `df.isnull().sum()`

7. What is the data type of the argument passed into the `pandas` method in the following expression?

   ```
   pd.DataFrame({"column_1": values_1, "column_2": values_2})
   ```
   D

   A. `pd.DataFrame` B. `keys` and `values` C. `str` and `np.array` D. `dict`

   E. None of the above because we can't tell what the data `types` are based on the information provided

8. Which of the following is the `pandas` expression that returns the first 5 rows of the last and third from the last columns (named `"z"` and `"x"` respectively) of `df` a `DataFrame` object? D

   A. `df[:5,[-3,-1]]` B. `df[:4,[-2,-0]]` C. `df.loc[:5,[-3,-1]]`
   D. `df.iloc[:5,[-3,-1]]` E. `df.iloc[:5,-2,-0]`

9. Suppose `df` is a `DataFrame` that has columns `Country`, `Athlete`, `Event`, and `Medal` where the `Medal` column has the values `"G"`, `"S"`, `"B"`, and `"None"`. Which of the following `pandas` expressions shows the values for `Country` and `Athlete` columns for the rows with the value `"G"` in the `Medal` column?

   A. `df[df["Medal"]=="G", ["Country","Athlete"]]`

   B. `df.loc[df.Medal=="G", "Country", "Athlete"]`

   C. `df.loc[df.Medal=="G", ["Country","Athlete"]]` C

   D. `df.iloc[df.Medal = "G", "Country", "Athlete"]`

   E. `df.iloc[df.Medal = "G", ["Country", "Athlete"]]`

10. Suppose `df` is a `DataFrame` that has columns `Country`, `Athlete`, `Event`, and `Medal` where the `Medal` column has the values `"G"`, `"S"`, `"B"`, and `"None"`. Which of the following `pandas` expressions returns ONLY the number of times the `Medal` column is `"G"` for each possible value in the `Country` column?

    A. `df[["Medal", "Country"]].groupby("Country").value_counts()`

    B. `df["Country"][Medal=="G"].groupby("Country").size()`

    C. `(df.groupby("Medal")["Country"]=="Gold").sum()`

    D. `df[df.Medal=="G"].groupby("Country").count()` D

    E. `df.groupby(["Country", df.Medal=="G"]).size()`

11. Suppose `df` is a `DataFrame` that has columns `Country`, `Athlete`, `Event`, and `Medal` where the `Medal` column has the values `"G"`, `"S"`, `"B"`, and `"None"`. Which of the following `pandas` expressions returns the rows of countries which won only bronze (`"B"`) or nothing (`"None"`)? D

    A. `df[ (df.Medal!="G")&(df.Medal!="S") & ((df.Medal=="B")|(df.Medal=="None")) ]`

    B. `df[ (df.Medal!="G") & (df.Medal!="S") ]`

    C. `df[ (df.Medal=="B") | (df.Medal=="None") ]`

    D. All of the above

    E. None of the above

12. Suppose `df` is a `DataFrame` that has columns `Country`, `Athlete`, `Event`, and `Medal` where the `Medal` column has the values `"G"`, `"S"`, `"B"`, and `"None"`. Which of the following is the most appropriate choice for the `type` of the `Medal` column? B

    A. `binary` B. `category` C. `int` D. `Object` E. `str`

13. Suppose `df` is a `DataFrame` that has columns `Country`, `Athlete`, `Event`, and `Medal` where the `Medal` column has the values `"G"`, `"S"`, `"B"`, and `"None"`. Which of the following `pandas` expressions is the best way to see which athletes in the `Athlete` column appear more than once in this data set? B

    A. `df.Athlete.value_counts().head()`

    B. `df.Athlete.value_counts()[df.Athlete.value_counts()>1]`

    C. `df.Athlete.value_counts().sort_values(ascending=True)`

    D. `df.Athlete.value_counts().sort_values(ascending=False)[:5]`

14. The theoretical *probability density function* of a *normal population* can be plotted on the basis of `stats.norm(loc=mu, scale=std).pdf(np.linspace(start,end,100))`.
What does `stats.norm(loc=mu scale=std).rvs(size=n)` do? B

    A. Visualizes a sample from the specified theoretical population

    B. Simulates a sample from the specified theoretical population

    C. Evaluates the probability density function of the specified theoretical population

    D. Checks the likelihood that an observed sample is from the specified theoretical population

15. Consider all the textbooks (including both suggested and required) for your courses. Describe two different populations from which these textbooks could be viewed as being samples from.
[2 points: 1 point for each of two required examples
and partial credit as described below and if a population example is somewhat ambiguous]
Good examples would be
1. The population of all textbooks at UofT this semester.
2. The population of textbooks selected for the classes classes you're enrolled in over time

{Students may alternatively describe a normal population; but, for full credit with this answer they must describe what variable the normal population is modeling; and, they can only use this kind of distribution example for just one of the two example populations they must provide.}

16. A *normal population* (often called a *normal distribution*) has two *parameters*, a *mean* μ (location) parameter and a *standard deviation* σ (scale) parameter. What calculations would be used to provide estimates of the population parameters? Provide both python code for a sample `x` stored as an `np.array` as well as the actual numeric calculations given in mathematical summation notation.
[4 points: 1 point for each of the following]

- `x.mean() # or something very similar -- HALF CREDIT for numeric calculation`
- `x.std() # or something very similar -- HALF CREDIT for numeric calculation`
- $\sum_{i=1}^{n} x_i/n$ – HALF CREDIT for no index notation or missing limits on summation
- $\sqrt{sum_{i=1}^{n}(x_i - \bar{x})^2/(n-1)}$ – MINUS QUARTER CREDIT for divide by n rather than $n-1$ and MINUS QUARTER CREDIT for no square root and MINUS QUARTER CREDIT for missing square on difference

17. What does the `nbins` parameter of the `plotly.express` function `px.histogram` do? Draw a couple examples demonstrating what the `nbins` parameter does.

[2 points: 1 point each for clearly demonstrating the following contrast]

- small number `nbins` histogram with few bins
- large number `nbins` histogram with many bins

DO NOT MARK ANSWERS ON YOUR SCANTRON FOR ITEMS #15-17: RESUME MARKING YOUR MULTIPLE CHOICE SCANTRON ANSWERS FROM ITEM #18

18. Which of the the following describes a *distribution* that is *left skewed*? A

    A. A big mass of larger numbers extending out towards smaller numbers rather than towards larger numbers
    B. A big mass of numbers that similarly extends towards both larger and smaller numbers
    C. A big mass of smaller numbers extending out towards larger numbers rather than towards smaller numbers
    D. Two big masses of numbers that are clearly distinguishable from one other other

19. Suppose `df` is a `pd.DataFrame` object with columns `x`, `values`, and `group`. What are the relative benefits of `px.histogram(df, x="x")` and `px.box(df, x="values", y="group")` that are specific and unique to each visualization, respectively? A

    A. Examining modality and n; Easily visualizing differences between empirical distributions
    B. Easily examining left and right skew; Examining modality and n
    C. Examining modality and n; Easily examining left and right skew
    D. Both show essentially the same information, so there's really no relative benefit either way

20. What's the difference between a *statistic* and a *parameter*? D

    A. One is $\bar{x}$ and the other is μ
    B. A *parameter* estimates a *statistic*
    C. A *statistic* is a number while a *parameter* is a greek letter
    D. They are the same but calculated with a *sample* or "theoretically" on the *population*
    E. A *statistic* is a mathematical calculation but a *parameter* is what is needed to describe and characterize the *empirical distribution* of a specific *sample*

21. The `df.shape` and `df.head()` utilize an *attribute* and *method* of the `df` object, respectively; or, similarly, `df.col.values` and `df.col.sum()` utilize an *attribute* and a *method* of an *attribute* of the same `df` object. Generally, which of the following is the best analogy for the difference between *attributes* and *methods*? A

    A. Viewing an *attribute* is like examining a *sample*; using a *method* is like calculating a *statistic*
    B. Viewing an *attribute* is like taking a *random sample*; using a *method* is like knowing a *population*
    C. Viewing an *attribute* is like calculating a *statistic*; using a *method* is like calculating a *parameter*
    D. Viewing an *attribute* is like applying a *function*; using a *method* is like indexing into an *array*

22. When doing *simulation*, the best practice is to use a *random seed* which makes the sequence of random numbers generated by the simulation *reproducible*. This is done so that every time you run the *simulation* with the same *random seed* you'll get the exact same results, which can help with *debugging* and *troubleshooting* and allows your results to be *reproducibly shared*. And of course, you can always choose another *seed* to change the random number sequence if you want to. Which of the following options best describes the code structure below introducing a *random seed* into this *simulation* creating a *bootstrap confidence interval* for a sample *statistic*? D

```
import numpy as np
reps = 10000 # number of bootstrap samples to collect
for i in range(reps):
    np.random.seed(130) # sets the random seed
    # now bootstrap sample, etc.
```

A. It will work for creating a *bootstrap confidence interval* AND makes it *reproducible*

B. It shows why you can't make *bootstrap confidence interval* construction *reproducible*

C. It will create a *bootstrap confidence interval* but fails to make "*reproducible*" *bootstrap samples*

D. To make this entire process *reproducible* `np.random.seed` should be moved above the `for loop`

23. The *boxplot* of a sample of size $n = 2$ numbers $x_{(1)} < x_{(2)}$ will be a box from the smaller number $x_{(1)}$ to the larger number $x_{(2)}$ because $x_{(1)}$ and $x_{(2)}$ are each closest to the 25th and 75th percentiles, and the *median* line will be then be in the middle at $\frac{x_{(1)}+x_{(2)}}{2}$. There won't be whiskers (usually of length of 1.5 times the *IQR*) since there's no data outside of the *IQR* range. What is the length of the 50% *bootstrap confidence interval* for the *population median* based on this sample of size 2? A

    Hint: Each bootstrap sample is equally like to be the smaller number twice, the larger number twice, the smaller number followed by the larger number, or the larger number followed by the smaller number...

A. 0 B. $x_{(2)} - x_{(1)}$ C. $\frac{x_{(1)}+x_{(2)}}{2} - x_{(1)}$ C. $x_{(2)} - \frac{x_{(1)}+x_{(2)}}{2}$

24. What is the correct interpretation of a 90% *confidence intervals*? B

A. The *parameter* of a *population* will sometimes be in the *confidence interval* 90% of the time, but it will change and leave the *confidence interval* randomly 10% of the time

B. There's a 90% chance the procedure will have "worked", and so there's a 90% chance the constructed 90% *confidence interval* did in fact capture the *true population parameter*

C. That exactly ninety (90) out of every one-hundred (100) 90% *confidence intervals* are guaranteed to capture the *true population parameter* since it's a *fixed value* that doesn't change

D. All of the above

E. None of the above

25. What's the most powerful way to decrease the length of a *confidence interval*? A

A. Work with a larger sample size n

B. Work with a smaller sample size n

C. Use a decreased confidence level, like 80% versus 99%

D. Use an increased confidence level, like 99% versus 80%

26. Suppose you have sample of data that is *right skewed* and has *outliers*. First, draw an example of a *boxplot* that would fit this data description, labeling all components of the *boxplot*. Then, indicate whether or not the sample *mean* or the sample *median* (50% percentile of the sample) will have a larger value by marking it on your *boxplot* relative to the *median*.
[3 points]

- (1/2) Q1/Q3 box
- (1/2) median AND mean to the right
- (1/2) whiskers
- (1/2) outliers
- (1) right skewed

27. Assuming the samples `x1` and `x2` are stored as `np.arrays`, provide the code to create a 90% *confidence interval* for the difference in the *population standard deviations* based on these samples.

```
from numpy.random import choice
from numpy import quantile
n1, n2 = len(x1), len(x2)
```

[4 points: should be a two sample difference in population standard deviations confidence interval
2 points for any correct bootstrapping and 1 more point for any correct two sample bootstrapping
If it's good but uses variance not standard deviation MINUS 1/2, and
If it's good but it's not a 90% CI another MINUS 1/2]

```
reps=10000 # any sensible/reasonable choice here is fine
stats = []
for i in range(reps):
    x1_ = choice(x1,n1)#replace=True is the default, so it's fine if they don't have it
    x2_ = choice(x2,n2)#replace=True is the default, so it's fine if they don't have it
    stats += [x1_.std() - x1_.std()] # either order is fine
 quantile(stats, (0.05, 0.95)
 # better if they preallocate and use an np.array
 #they can also do this with pandas .sample(..., replace=True) but need 'replace=True'!!
```

DO NOT MARK ANSWERS ON YOUR SCANTRON FOR ITEMS #26-27: RESUME MARKING YOUR MULTIPLE CHOICE SCANTRON ANSWERS FROM ITEM #28

28. What is the fundamental concept of bootstrapping? C

    A. Using `np.random.seed(...)`
    B. Creating and interpreting confidence intervals
    C. Presuming the sample can be used in place of the population
    D. Sampling without replacement to create the sampling distribution of the test statistic
    E. Using the distribution of the population of the data to understand sampling variability

29. What does `np.random.choice([1,2,3]), replace=False, size=4)` produce if we instead use `replace=True`? B

    A. It will produce a bootstrap sample
    B. It will flip a "fair" three-sided coin four times
    C. Nothing, the code will not run and will produce an error
    D. It will return a shuffled version of original list of data `[1,2,3]`
    E. None of the above

30. Why must *bootstrapping* be based on sampling *with replacement*? E

    A. So that the sample size of the *bootstrap* sample is the same as the *original* sample
    B. So the values in *bootstrap* sample are not exactly the same as the *original* sample
    C. It allows us to *simulate sampling variability* in the *statistic* for the *same sample size as the original sample* without knowing the *population* and only knowing the *original sample* itself
    D. Both A and B
    E. Both B and C

31. Which of the following most specifically describes the *variability* of a *statistic*? D

    A. The distribution
    B. The empirical distribution of the sample
    C. The population distribution
    D. The so-called sampling distribution
    E. None of these options

32. Which of the following is the best definition of a *p-value*? E

    A. The probability of making a *Type I* error
    B. The probability that the *null hypothesis* is true
    C. The *parameter* of a *binomial distribution* specifying the chance of a "success"
    D. The probability that the *parameter* equals the value hypothesized by the *null hypothesis*
    E. The probability of getting a test statistic as or more extreme than the *observed test statistic* if the *null hypothesis* is true

33. For sample `x` stored as an `np.array`, write code to simulate the *p-value* for the *null hypothesis*

$$H_0 : \sigma_x = 1 \text{ with } \texttt{x} \text{ a sample of size } n \text{ from a } \mathcal{N}(0, \sigma) \text{ population}$$

```
from scipy.stats import norm
```

[3 points:
looping and simulated test statistic structure worth 1 point
correct simulation and test statistic calculation worth 1 point
p-value correctly calculated
half credit for each of these for "not quite right but getting close"]

```
stats = [] # better to do this preallocation
reps=10000 # any sensible/reasonable choice here is fine
for i in range(reps):
    stats += [norm(loc=0,std=1).rvs(n).std()]
(abs(np.array(stats)-1)>=abs(x.std()-1)).sum()/reps # lose some points without np.array
```

34. The *binomial probability mass function* (often referred to as the *binomial distribution*)

$$Pr(X = k) = \left(\begin{array}{c} n \\ k \end{array} \right) p^k n^{n-k}$$

is a mathematical expression giving the probability of k "successes" out of n "attempts".

> The $\left(\begin{array}{c} n \\ k \end{array} \right)$ "n choose k" notation counts the unique ways to chose k out of n things based on the order in which the k things sequentially appear, and it is a *combinatorics* calculation that is required so the binomial expression provides the correct probability values.

The *binomial distribution* is a model of the population of the number of "successes" k there are out of n "attempts", assuming the chance of "success" doesn't change depending on previous or future outcomes. The *binomial distribution* doesn't require any "distributional" assumptions, just the assumption that the performance is consistent. For example, we don't need to say the "shape is normally distributed" or something along those lines to derive the *binomial distribution*.

We could thus model your (assumed "pretty good") ability to "make basketball shots" through the

null hypothesis H_0 : the chance you make each shot is 70% (so $p = 0.7$ for the *binomial model*)

The probabilities of the various number of k shots you would make out of $n = 10$ tries (possibly 0 up to 10) under the consistent performance assumption of *binomial model* then are

```
>>> from scipy import stats; import numpy as np
>>> stats.binom(p=0.7,n=10).pmf(np.linspace(0,10,11,dtype=int))
array([0.000005905, 0.000137781, 0.0014467005, 0.009001692,
       0.036756909, 0.102919345, 0.2001209490, 0.266827932,
       0.233474441, 0.121060821, 0.0282475249]) # these sum to 1
```

Draw a *barplot* of this *binomial distribution* then circle the probability or probabilities to be added together to calculate the theoretical p-value for $k = 10$ made shots out of $n = 10$ attempts for

null hypothesis H_0 : the chance you make each shot is 70% (so $p = 0.7$ for the *binomial model*)

[4 points:]

- 2 points: bar plot should have the heights right rank order for each probability, but the heights need only be approximately correct
  ⋯ • 1 point (partial credit) if there's not 11 bins from 0 to 10 but it's otherwise pretty close
- 2 points: p-value is >= 10 for one-sided and also <= 4 for two-sided (10-7=3 and 4-7=-3); so, 2.82475249e-02
  [+ 5.90490000e-06 + 1.37781000e-04 + 1.44670050e-03 + 9.00169200e-03 + 3.67569090e-02]
  ⋯ • no partial credit for circling k=10 and k=0; and, if only k=10 is circled the student gets 1 point (partial credit) and only gets full credit if they clearly specify "one sided test"
  ⋯ • AKA students should circle both the left and the right of 7 symmetrically unless they specify they're doing a one-sided test

DO NOT MARK ANSWERS ON YOUR SCANTRON FOR ITEMS #33-34: RESUME MARKING YOUR MULTIPLE CHOICE SCANTRON ANSWERS FROM ITEM #35

35. Use the table below to characterize the strength evidence against H_0 for the previous question if "as or more extreme" refers to both above and below the performance expected by the *null hypothesis*.
B

	p-value	Evidence against the Null Hypothesis
A.	$p > 0.10$	No Evidence
B.	$0.05 < p < 0.10$	Weak Evidence
C.	$0.01 < p < 0.05$	Moderate Evidence
D.	$0.001 < p < 0.01$	Strong Evidence
E.	$p < 0.001$	Very Strong Evidence

A. No Evidence B. Weak Evidence C. Moderate Evidence D. Strong Evidence E. Very Strong Evidence

36. What would the strength evidence against H_0 be for the previous question if "as or more extreme" refers only to performances that are above the expectations of the *null hypothesis*? C

A. No Evidence B. Weak Evidence C. Moderate Evidence D. Strong Evidence E. Very Strong Evidence

37. Making 10 shots out of 10 attempts as considered in the previous problems is "perfect accuracy". Should we therefore conclude that "you shoot with perfect accuracy"? C

A. Yes, the evidence is very strongly in favor of this conclusion
B. Yes, so long as the *p-value* is based on the *one-sided* version of the calculation
C. Maybe, but a larger sample size would likely strengthen our confidence one way or another
D. No, it's never the case that it's possible to have "perfect accuracy"
E. No, at the current sample size the hypothesis test was able to prove that 10 out of 10 is not really "perfect accuracy"

38. Would it be helpful to increase the sample size in the context of the previous problems? D

A. No, the "10 out of 10" evidence is already sufficiently strong and does not require increased n
B. No, because the p-values can already be calculated to determine whether or not H_0 is correct
C. Yes, because then the p-value could be more accurately estimated
D. Yes, because then the variance of the sampling distribution of the test statistic under the assumption of H_0 would be reduced, which would decrease the p-value of the observed test statistic
E. Yes, because then the variance of the sampling distribution of the test statistic under the assumption of H_0 would be reduced, which would increase the p-value of the observed test statistic

39. What's the difference between a simulated p-value and a p-value based on the *binomial distribution*, such as those considered in the previous few problems? Complete the following sentence. B

The *simulated* p-value approximation of the p-value based on the *binomial distribution*...

A. improves as the sample size n increases
B. improves as more simulated test statistics are used to estimate the p-value
C. is based on different assumptions which is what makes it different
D. is never as accurate and so is never as useful as the real p-value
E. is not possible to create since test statistics cannot be simulated under H_0 in this context

40. What distinguishes a *parametric* p-value from a *nonparametric* p-value? C

A. Assumptions about parameters such as $H_0 : \mu = 0$ or that two samples have the same medians
B. Whether a sample is randomly drawn from the population or if it has some systematic bias
C. Assumptions about the distribution of the data such as that it's from a normal population
D. Whether it can be approximated using *simulation* or only be calculated *theoretically*
E. Whether or not the analysis is based on one sample (one-sided) or two samples (two-sided) test

41. Propose theoretical populations for first (a) the (*continuous* valued) weight of a book and second (b) for the (*discrete* valued) number of books that are bought off of a display shelf with five books. Then, provide (c) an alternative approach to providing inference based on samples that might be associated with these "populations" in the event that these theoretical populations aren't actually very accurate specifications of the actual populations the samples are drawn from.

[3 points: 1 point for good answers close to each of the following]

(a) normal(mu=10oz?, std=3oz?) – questions marks because anything remotely sensible is fine

(b) binomial(n=5,p=0.1) – again, anything for p not 0 or 1 is fine, but n and binom are needed
students can alternatively provide some variation on, e.g.,
`np.random.choice([0,1,2,3,4,5],p=[.3,.3,.2,.1,.1])`

(c) Bootstrap! Pretend the sample is a good representative of the population...
it's less technically clear, but I like to describe this as "make a histogram of the empirical distribution of the sample" and pretend that's the population distribution!"

42. Write out the *null hypothesis* that there is no difference in the *standard deviations* σ_1 and σ_2 of the respective *populations* of two samples x_1 and x_2. Then, assuming the samples `x1` and `x2` are stored as `np.array`s, provide the code to create a *p-value* based on a *permutation test* of this *null hypothesis*.

[4 points:
1 point: hypothesis specification (as below, okay if includes additional information like sample sizes)
1 point: shuffling group labels (or numeric values, either is fine)
1 point: calculating statistic correctly (e.g., as demonstrated with groupby)
1 point: p-value calculation (including for loop and simulated/permutation statistic collection]

$$H_0 : \sigma_1 = \sigma_2$$

```
import pandas as pd
from numpy import diff
df = pd.DataFrame({"x": list(x1)+list(x2), "s": list(1+0*x1)+list(2+0*x2)})
```

```
stats = [] # better if they pre-allocate, but list appending is fine
reps=10000 # any sensible/reasonable choice here is fine
for i in range(reps):
    shuf = df.s.sample(frac=1, replace=False) # pd default replace=False is not needed
    # or equivalent for the of the above      # size must be frac=1 or df.shape[0]
    stats += [np.diff(df.groupby(shuf).std())[0]] # or equivalent
(abs(diff(df.groupby(df.s).std())[0]) >= abs(np.array(stats))).sum()/reps
```

DO NOT MARK ANSWERS ON YOUR SCANTRON FOR ITEMS #41-42: RESUME MARKING YOUR MULTIPLE CHOICE SCANTRON ANSWERS FROM ITEM #43

43. Is the *p-value* in the previous problem *theoretical* or *simulated*? Is it *parametric* or *nonparametric*?

    A. theoretical parametric
    B. simulated parametric
    C. theoretical nonparametric
    D. simulated nonparametric D
    E. none of the above

44. Illustrate survivor bias with an example that's not related to WW2 or music.

    [2 points:
    1 point: a legitimate example (can give 1/2 point partial credit on a subjective basis)
    1 point: well-written (can give 1/2 point partial credit on a subjective basis)

    This example is anything where the observations at the end have been filtered through the process causing them to be a biased representative of what is actually the full spectrum of experience within the population of interest.

    So, e.g., the planes that come back are the ones that survived, which means we don't see the planes that got shot down; or the songs that a remembered from old times are the ones that have been generally more liked (which could be contrasted with current songs which have gone through the same "test of time" process and survived).

    Another example I'm thinking about could be university students by the time they get to the end do not represent the full experience of all university students, some of whom transferred, or others who dropped out, etc.

    Maybe another example would be sampling restaurants, which would be comprised of many of which would have also withstood the "test of time".

    Anyway, these sorts of things seem like fair examples to me. Etc.

45. Name three *statistics* that are computed for the *numeric* columns of a `pandas` object `df` of type `pd.DataFrame` when `df.describe()` is run, and write how they could be computed for an `np.array` object `x` using the *methods* of the `x` object.

    [3 points:
    1/2 point each for naming three
    1/2 point each for the corresponding numpy method

    - standard deviation / `.std()` or also fine is variance / `.var()`
    - mean / `.mean()`
    - Q1,Q2,Q3 with `.quantile([0.25,0.5,0.75])` `xpercent` quantiles with `.quantile(xpercent)` and just one of these is fine and if it's written in terms of `..percentile()` that's fine too
      it's also fine if it's written as `95%` or `.95` for either `.quantile` or `..percentile()`
      we don't need to worry about specific careful details there for this marking
    - min / `.min()`, max / `.max()`, count / `.shape` ¡- must be an *attribute* if used as an example