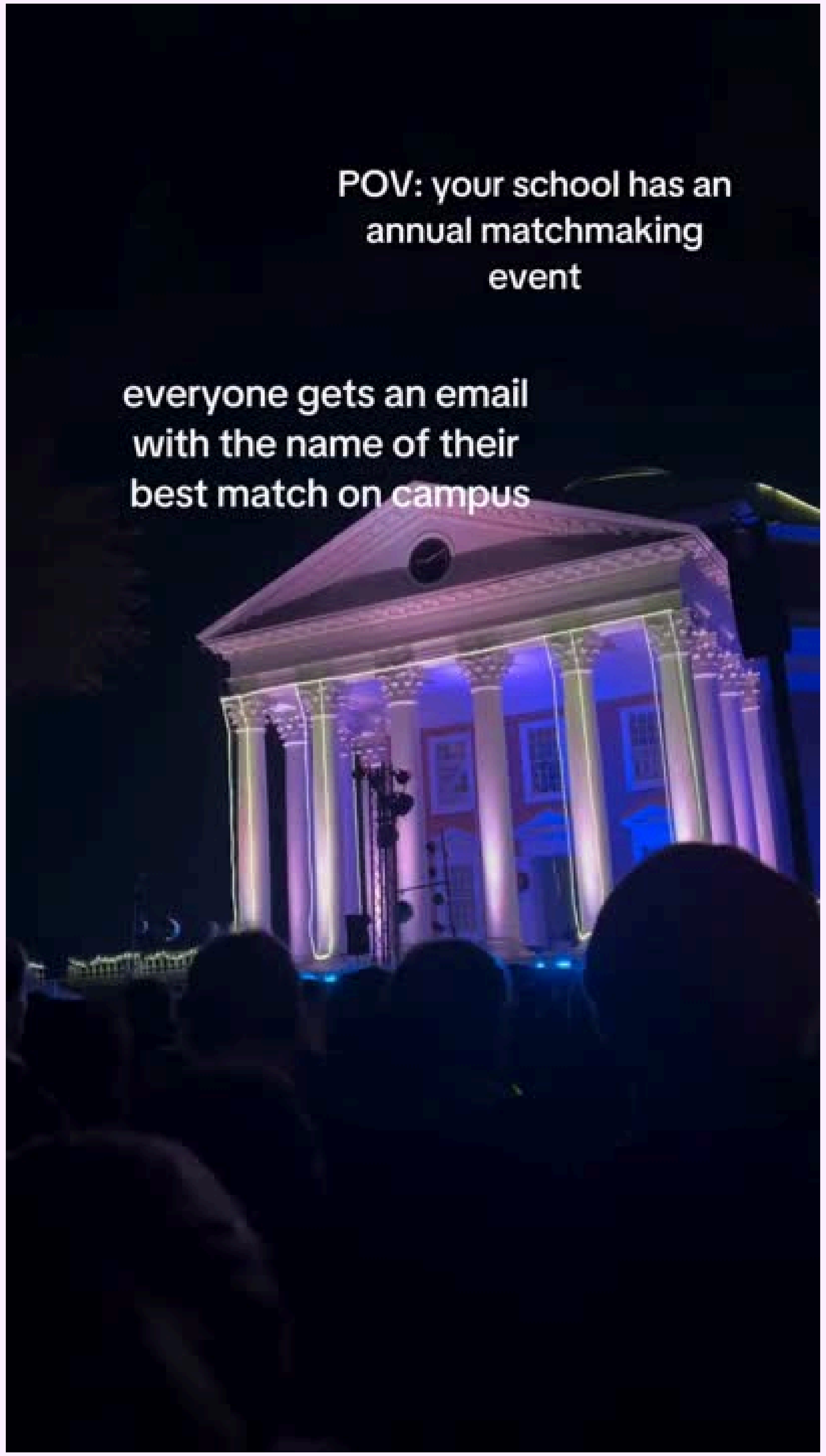
Friendship Finder

By
Arjun Maneesh
Agarwal


POV: your school has an annual matchmaking event
everyone gets an email with the name of their best match on campus

One of my school friends had liked this reel.

And I went like: How hard could it be? I’ll make it!

That was exactly 9 months ago.

We will see algorithms, Haskell code math, stats and some psychology. Hopefully, it will be fun.

02

PART 02

Introduction to Matching Theory

The Prize in Economic Sciences 2012

Alvin E. Roth
Harvard University, Cambridge, MA, USA and Harvard Business School, Boston, MA, USA

Lloyd S. Shapley
University of California, Los Angeles, CA, USA

"för teorin om stabila allokeringar och för utformning av marknadsinstitutioner i praktiken"

"for the theory of stable allocations and the practice of market design"

A Slightly Easier Problem

Given n men and n women, who are all heterosexual and have a complete preference ordering over the others, can we match them nicely?

Um, but what is a nice matching?

Well, it could mean many things.

- Popular matching
- Maximin matching
- Pareto matching

Unfortunately, all of these are NP complete to find. However, there is a type of matching which could work...

That is?

Stable Matching

A matching $\mathcal{M}$ on $M \cup W$ is stable if

$$\forall m \in M, \forall w \in W; (m, w'), (m', w) \in \mathcal{M}$$

$$w' \succ_m w \text{ and } m' \succ_w m$$

And how do we find this matching?

We will use something called the method of deferred acceptance or Gale-Sharpley Algorithm!

Gale-Sharpley Algorithm

Consider a party with 4 males and 3 females. Let's name these people Sheldon, Leonard, Raj and Howard; and Penny, Amy and Bernadette.

Please note, despite whatever the preferences look like; This table has nothing to do with any heavily copyrighted show.

	1st Choice	2nd Choice	3rd Choice
Raj	Penny	Amy	Bernadette
Sheldon	Amy	Penny	Bernadette
Leonard	Penny	Bernadette	Amy
Howard	Penny	Bernadette	Amy

	1st Choice	2nd Choice	3rd Choice	4th Choice
Penny	Leonard	Sheldon	Raj	Howard
Amy	Sheldon	Raj	Leonard	Howard
Bernadette	Howard	Raj	Leonard	Sheldon

Gale-Sharpley Algorithm

Each round:

1. Boys who are without a provisional match propose to the their top girl who is yet to reject them
2. Each Girl rejects all but the favourite offer received that round + her provisional match.
3. Each Boy crosses off the rejecting girls from their list.
4. Repeat till everyone has a provisional match. These are final matches.

	1st Choice	2nd Choice	3rd Choice
Raj	~~Penny~~	Amy	Bernadette
Sheldon	$\boxed{\text{Amy}}$	Penny	Bernadette
Leonard	$\boxed{\text{Penny}}$	Bernadette	Amy
Howard	~~Penny~~	Bernadette	Amy

	1st Choice	2nd Choice	3rd Choice	4th Choice
Penny	$\boxed{\text{Leonard}}$	Sheldon	$\underline{\text{Raj}}$	$\underline{\text{Howard}}$
Amy	$\boxed{\text{Sheldon}}$	Raj	Leonard	Howard
Bernadette	Howard	Raj	Leonard	Sheldon

Gale-Sharpley Algorithm

Each round:

1. Boys who are without a provisional match propose to the their top girl who is yet to reject them
2. Each Girl rejects all but the favourite offer received that round + her provisional match.
3. Each Boy crosses off the rejecting girls from their list.
4. Repeat till everyone has a provisional match. These are final matches.

	1st Choice	2nd Choice	3rd Choice
Raj	~~Penny~~	~~Amy~~	Bernadette
Sheldon	$\boxed{\text{Amy}}$	Penny	Bernadette
Leonard	$\boxed{\text{Penny}}$	Bernadette	Amy
Howard	~~Penny~~	$\boxed{\text{Bernadette}}$	Amy

	1st Choice	2nd Choice	3rd Choice	4th Choice
Penny	$\boxed{\text{Leonard}}$	Sheldon	Raj	Howard
Amy	$\boxed{\text{Sheldon}}$	$\underline{\text{Raj}}$	Leonard	Howard
Bernadette	$\boxed{\text{Howard}}$	Raj	Leonard	Sheldon

Gale-Sharpley Algorithm

Each round:

1. Boys who are without a provisional match propose to the their top girl who is yet to reject them
2. Each Girl rejects all but the favourite offer received that round + her provisional match.
3. Each Boy crosses off the rejecting girls from their list.
4. Repeat till everyone has a provisional match. These are final matches.

	1st Choice	2nd Choice	3rd Choice
Raj	~~Penny~~	~~Amy~~	~~Bernadette~~
Sheldon	$\boxed{\text{Amy}}$	Penny	Bernadette
Leonard	$\boxed{\text{Penny}}$	Bernadette	Amy
Howard	~~Penny~~	$\boxed{\text{Bernadette}}$	Amy

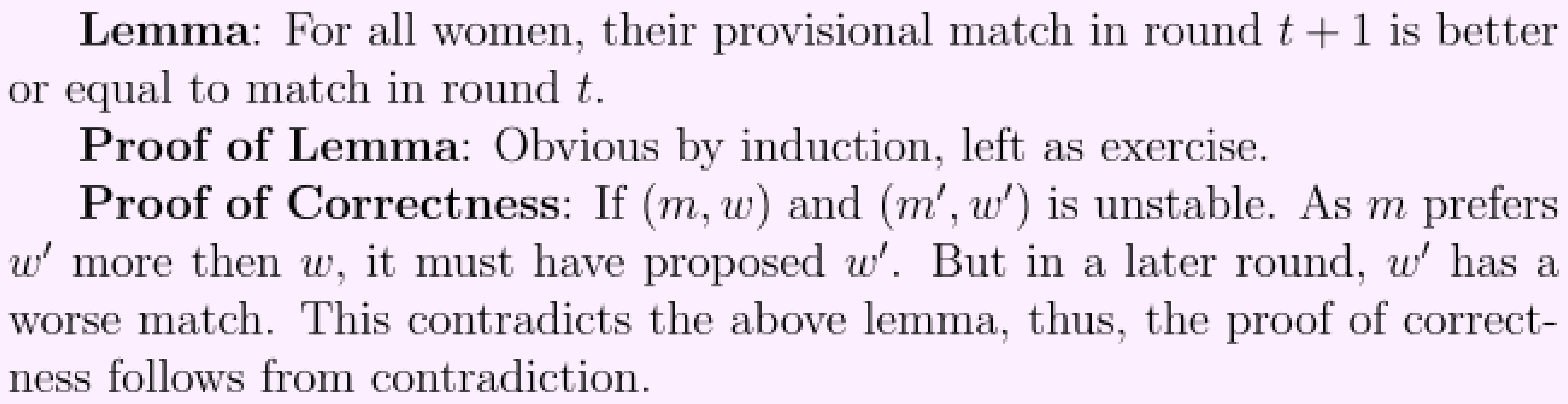
	1st Choice	2nd Choice	3rd Choice	4th Choice
Penny	$\boxed{\text{Leonard}}$	Sheldon	Raj	Howard
Amy	$\boxed{\text{Sheldon}}$	Raj	Leonard	Howard
Bernadette	$\boxed{\text{Howard}}$	$\underline{\text{Raj}}$	Leonard	Sheldon

Analysis of Algorithm

It is clear that at most n*m proposals can be made and hence, O(n*m) updates need to be made.

Hence, given equal number of men and women, we make O(n^2) proposals and the algorithm runs in quadratic time (with the appropriate data structures).

What about correctness? LaTex time!



Lemma: For all women, their provisional match in round $t+1$ is better or equal to match in round t.

Proof of Lemma: Obvious by induction, left as exercise.

Proof of Correctness: If (m, w) and (m', w') is unstable. As m prefers w' more then w, it must have proposed w'. But in a later round, w' has a worse match. This contradicts the above lemma, thus, the proof of correctness follows from contradiction.

The Actual Problem

Um, but what do we do when gender is not a constraint?

Let's reframe the problem. Given 2n people with complete preference over the others, how do we find a stable matching?

What does Stable mean in this context? We used gender in the last definition!

Leaving the formalization to you, we say the matching is stable if no two participants prefer the other over their match.

The Actual Problem

This doesn't even exist always!

Consider

A: B > C > D

B: C > D > A

C: D > A > B

D:A > B > C

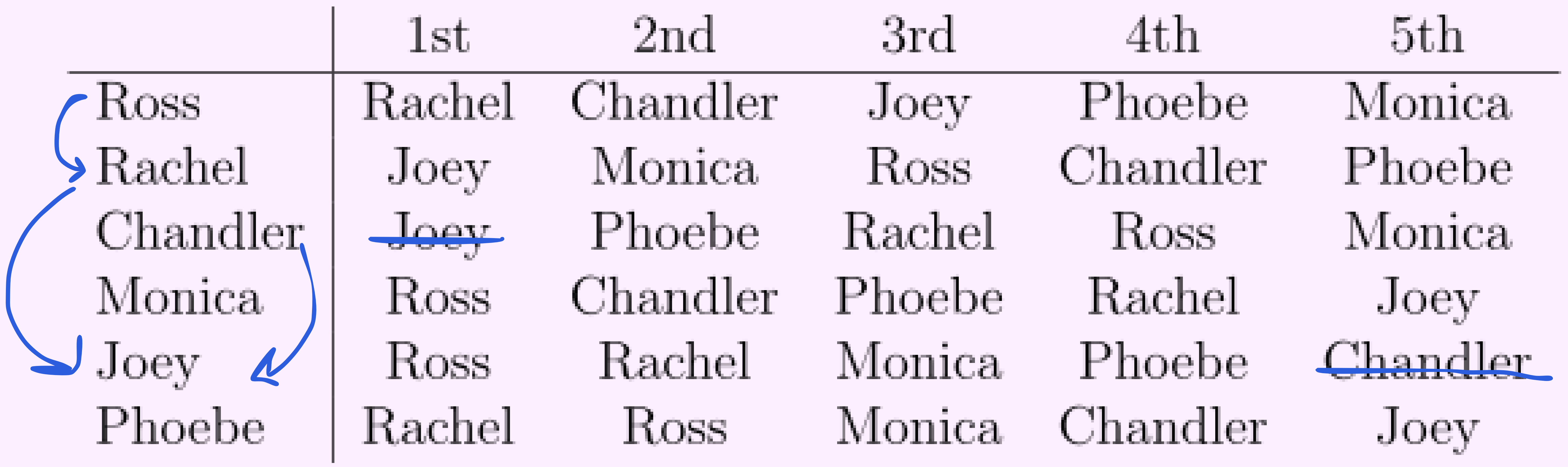
This is what makes the problem harder! However, the deferred acceptance scheme with some changes can find a solution or indicate that it doesn't exist.

Irving's Algorithm

Found by Robert Irving in 1985.

The first phase of this algorithm is similar to Gale-Sharpley. Let's just look at it!

Again, any resemblance to any copy-righted material is purely coincidental.



	1st	2nd	3rd	4th	5th
Ross	Rachel	Chandler	Joey	Phoebe	Monica
Rachel	Joey	Monica	Ross	Chandler	Phoebe
Chandler	~~Joey~~	Phoebe	Rachel	Ross	Monica
Monica	Ross	Chandler	Phoebe	Rachel	Joey
Joey	Ross	Rachel	Monica	Phoebe	~~Chandler~~
Phoebe	Rachel	Ross	Monica	Chandler	Joey

Irving's Algorithm

Found by Robert Irving in 1985.

The first phase of this algorithm is similar to Gale-Sharpley. Let's just look at it!

Again, any resemblance to any copy-righted material is purely coincidental.

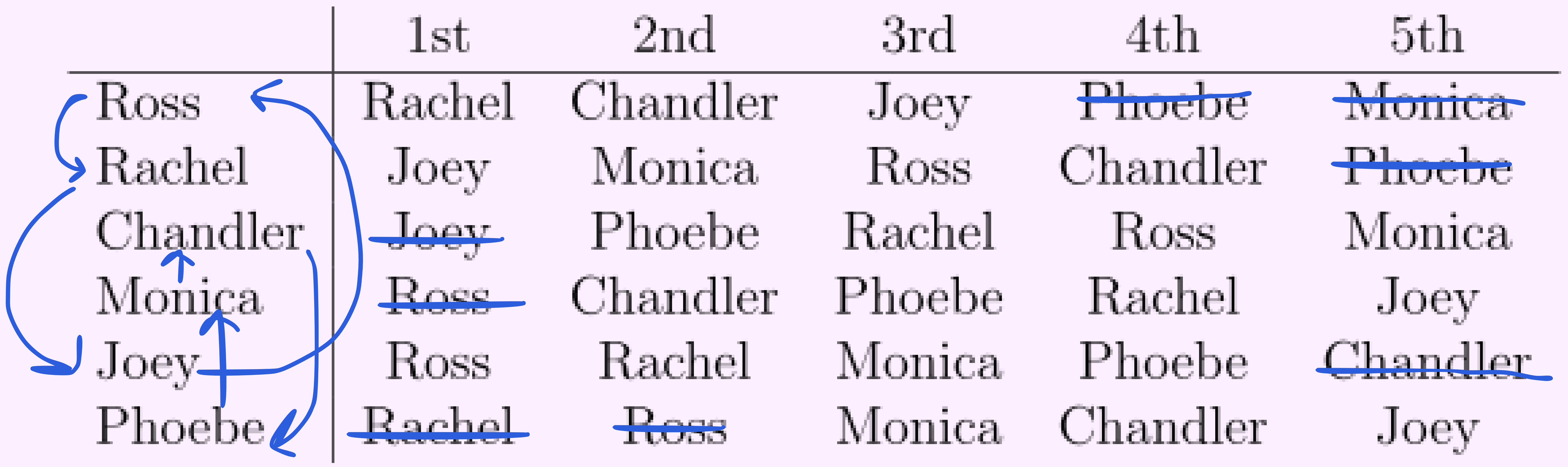
	1st	2nd	3rd	4th	5th
Ross	Rachel	Chandler	Joey	Phoebe	~~Monica~~
Rachel	Joey	Monica	Ross	Chandler	Phoebe
Chandler	~~Joey~~	Phoebe	Rachel	Ross	Monica
Monica	~~Ross~~	Chandler	Phoebe	Rachel	Joey
Joey	Ross	Rachel	Monica	Phoebe	~~Chandler~~
Phoebe	Rachel	Ross	Monica	Chandler	Joey

Irving's Algorithm

Found by Robert Irving in 1985.

The first phase of this algorithm is similar to Gale-Sharpley. Let's just look at it!

Again, any resemblance to any copy-righted material is purely coincidental.



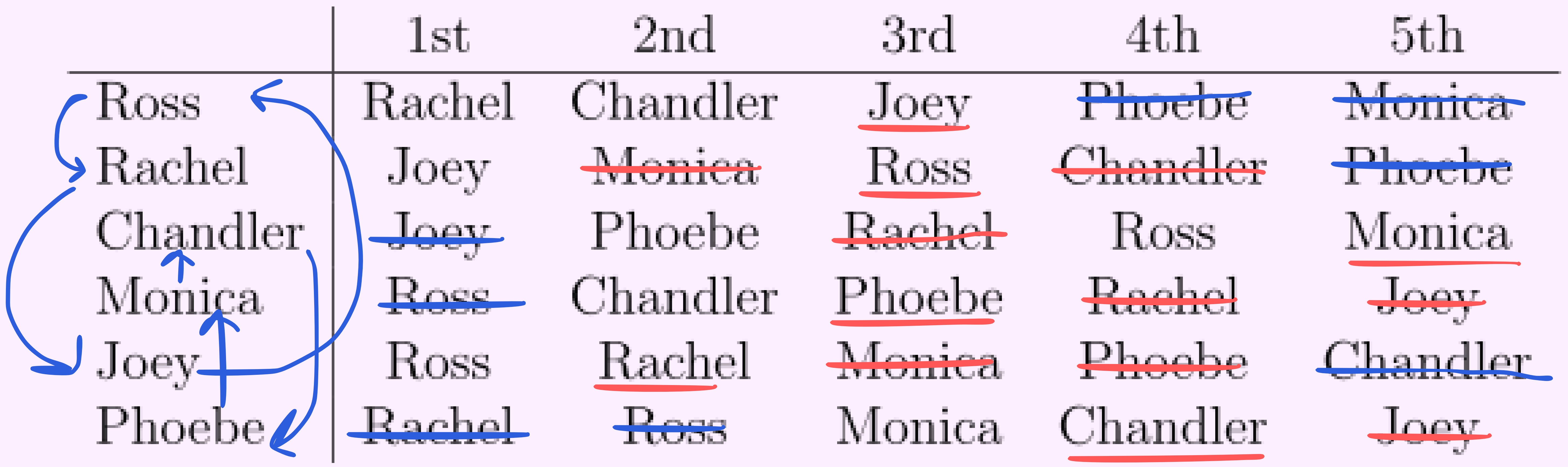
	1st	2nd	3rd	4th	5th
Ross	Rachel	Chandler	Joey	~~Phoebe~~	~~Monica~~
Rachel	Joey	Monica	Ross	Chandler	~~Phoebe~~
Chandler	~~Joey~~	Phoebe	Rachel	Ross	Monica
Monica	~~Ross~~	Chandler	Phoebe	Rachel	Joey
Joey	Ross	Rachel	Monica	Phoebe	~~Chandler~~
Phoebe	~~Rachel~~	~~Ross~~	Monica	Chandler	Joey

Irving's Algorithm

Found by Robert Irving in 1985.

The second phase of this algorithm is rejecting all the people below your current proposal.

We'll move to a reduced list post that. There is a check for stability at that point.



	1st	2nd	3rd	4th	5th
Ross	Rachel	Chandler	Joey	~~Phoebe~~	~~Monica~~
Rachel	Joey	~~Monica~~	Ross	~~Chandler~~	~~Phoebe~~
Chandler	~~Joey~~	Phoebe	~~Rachel~~	Ross	Monica
Monica	~~Ross~~	Chandler	Phoebe	~~Rachel~~	~~Joey~~
Joey	Ross	Rachel	~~Monica~~	~~Phoebe~~	~~Chandler~~
Phoebe	~~Rachel~~	~~Ross~~	Monica	Chandler	~~Joey~~

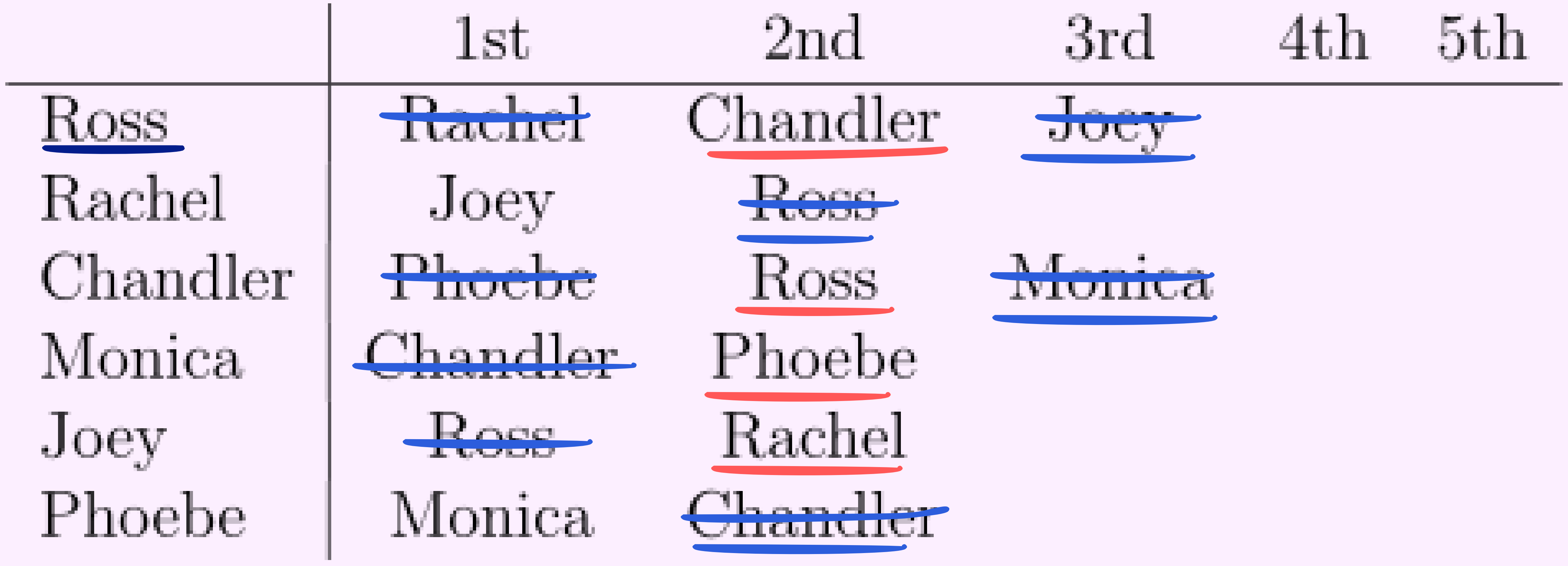
Irving's Algorithm

Notice, any singleton lists indicate their only possible matches! So we try to get to singletons.

The Third phase of this algorithm is eliminating the cycles. Notice the cycle:

$$\text{Ross} \rightarrow \text{Rachel} \rightarrow \text{Joey} \rightarrow \text{Ross}$$

So we can try to get their by removing the last elements in a stable manner. If any of the list goes to 0, no stable matching exists!



	1st	2nd	3rd	4th	5th
Ross	~~Rachel~~	Chandler	~~Joey~~		
Rachel	Joey	~~Ross~~			
Chandler	~~Phoebe~~	Ross	~~Monica~~		
Monica	~~Chandler~~	Phoebe			
Joey	~~Ross~~	Rachel			
Phoebe	Monica	~~Chandler~~			

Conclusion

Proof of correctness and complexity are both much harder in this case, but the idea's are roughly the same. It is left as exercise to do so.

The complexity comes out to be $O(n^2)$.

But this is an extremely complicated algorithm! The official Python implementation is 75+ lines removing comments and empty lines. Same can be said for the Java, C++, R implementations.

So why use Haskell for this?

03

PART 03

Why Haskell?

LEFT
EXIT 12
Using Python Library
Implement in Haskell
EAST ST WESTWOOD
Me
imgflip.com

Algo in Python
Algo in Haskell
imgflip.com

Jokes aside

Haskell (Functional Programming as a whole) is a much better language for dealing with complex algorithms. The ease of composing functions combined with higher order functions like fold, map, zip etc make a lot of work easier.

I will also point out that I will read the next pieces of code after almost an year. I probably will understand all of it thanks to Haskell, but if I don’t; that me being a noob and should not be held against Haskell.

Phase 1

```haskell
irv1 :: [(String, [String])] -> ([(String,String)], [(String, [String])])
irv1 people = go (M.fromList people) M.empty (map fst people) where
    go pref ans [] = (M.toList ans, M.toList pref)
    go pref ans (b:bs) = case M.findWithDefault [] b pref of
        [] -> go (M.delete b pref) ans bs
        (p:ps) ->
            let prefP = M.findWithDefault [] p pref
            in if M.notMember p ans then go pref (M.insert p b ans) bs else
                let b' = M.findWithDefault "none" p ans
                    pred = elem b (takeWhile (/= b') prefP)
                     -- This is true if b is better than b' and false otherwise
                    newB' = filter (/= p) (M.findWithDefault [] b' pref) -- if p rejects b'
                    newB = filter (/= p) (M.findWithDefault [] b pref) -- if p rejects b
                    newP' = filter (/= b) (M.findWithDefault [] p pref)  -- if p rejects b
                    newP = filter (/= b') (M.findWithDefault [] p pref) -- if p rejects b'
                    newPref = M.insert b' newB' $ M.insert p newP pref
                    newPref' = M.insert b newB $ M.insert p newP' pref
                in if pred then go newPref (M.insert p b ans) (b':bs) else go newPref' ans (b:bs)
                -- If pred == True; update preference of p and b' to remove each other; this is reflected in newPref
                -- If pred == False; update preference of p and b; this is reflected in newPref'
```

Phase 2

```haskell
rejection :: Eq a => [a] -> a -> ([a],[a])
rejection [] _ = ([],[])
rejection (x:xs) y = if x==y then ([x], xs) else (x:l, r) where (l,r) =rejection xs y

irv2 :: ([(String,String)], [(String, [String])]) -> [(String, [String])]
irv2 (pairs, pref) = go (M.fromList pref) pairs where
    go ans [] = M.toList ans
    go ans ((x,y):xs) =
        let xPref = M.findWithDefault [] x ans
            (new, rejects) = rejection xPref y
        in go (removeRejects (M.insert x new ans) x rejects) xs

    removeRejects :: M.Map String [String] -> String -> [String] -> M.Map String [String]
    removeRejects ans _ [] = ans
    removeRejects ans x (y:ys) = removeRejects newAns x ys where
        prefY = M.findWithDefault [] y ans
        newAns = M.insert y (filter (/= x) prefY) ans
```

Some Helpers!

```haskell
tableStable :: [(String, [String])] -> Bool
tableStable l = [] `notElem` map snd l

areWeDone :: [(String, [String])] -> Bool
areWeDone = all ((\x -> length x == 1) . snd)

detectCycle :: [(String, [String])] -> [(String, String)]
detectCycle pref = go (M.fromList pref) [x] [] x
  where
    x = fst $ head pref
    go pMap ll@(l:ls) rl name =
      let lPref = M.findWithDefault [] l pMap
          newRmember = if length lPref > 1 then lPref !! 1 else ""
          rPref = M.findWithDefault [] newRmember pMap
          newLmember = if null rPref then "" else last rPref
      in if newLmember == name
         then zip (newLmember:ll) (newRmember:rl)
         else go pMap (newLmember:ll) (newRmember:rl) name
```

Phase 3

```haskell
irv3 :: [(String, [String])] -> [(String, String)]
irv3 pref
    | areWeDone pref = map (\(x,y) -> (x, head y)) pref
    | not (tableStable pref) = error "No stable pairing found!"
    | otherwise = irv3 (go (M.fromList pref) (detectCycle pref))
  where
    go ans [] = M.toList ans
    go ans ((x,y):ys) =
        let xPref = M.findWithDefault [] x ans
            yPref = M.findWithDefault [] y ans
            xPref' = filter (/= y) xPref
            yPref' = filter (/= x) yPref
            newAns = M.insert x xPref' $ M.insert y yPref' ans
        in go newAns ys
```

Putting Stuff Together!

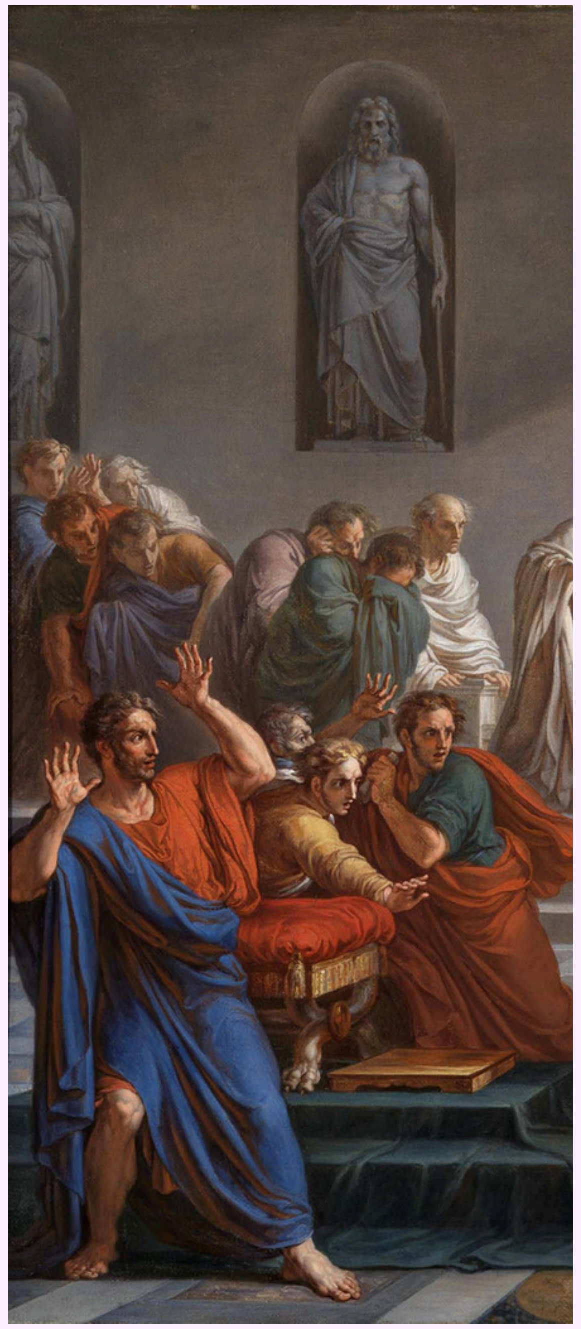
```
roomies :: [(String, [String])] -> [(String, String)]
roomies list = nubBy (\(a,b) (c,d) -> ((a,b) == (c,d)) || ((a,b) == (d,c))) $ irv3.irv2.irv1 $ list
```

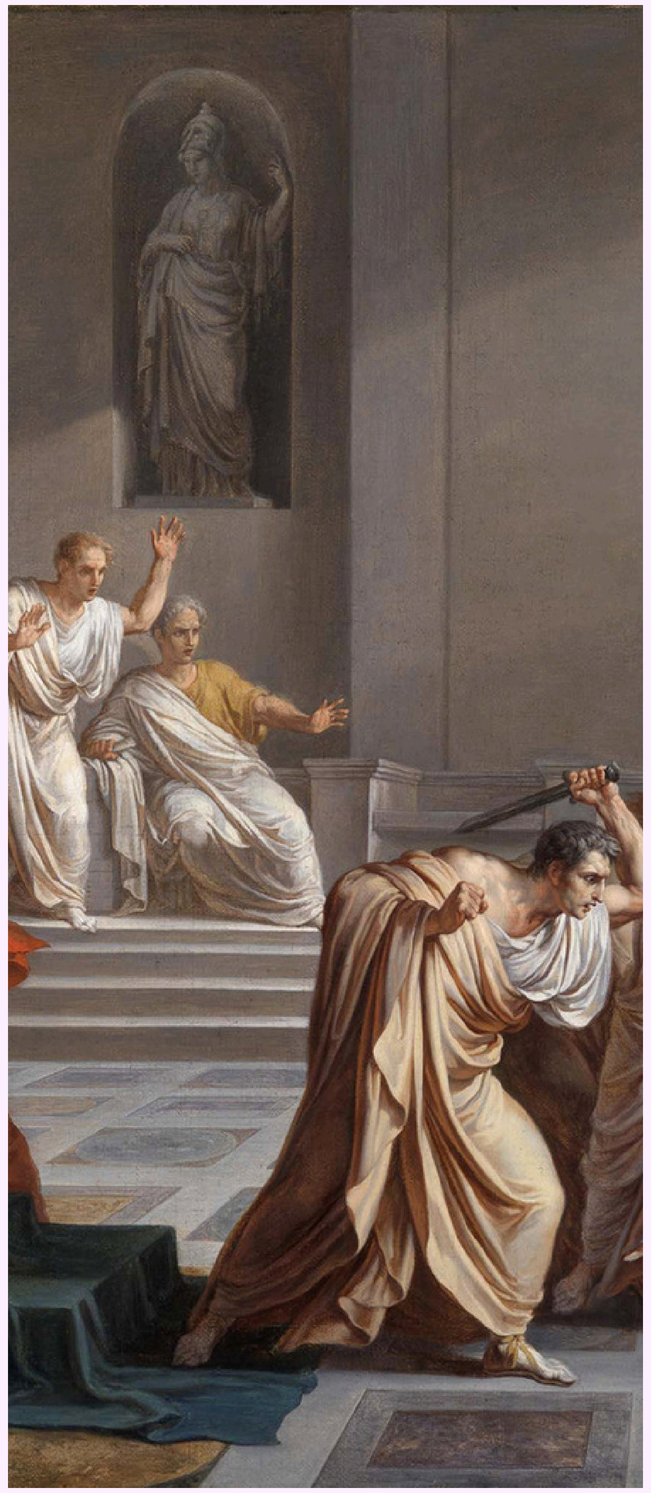
04

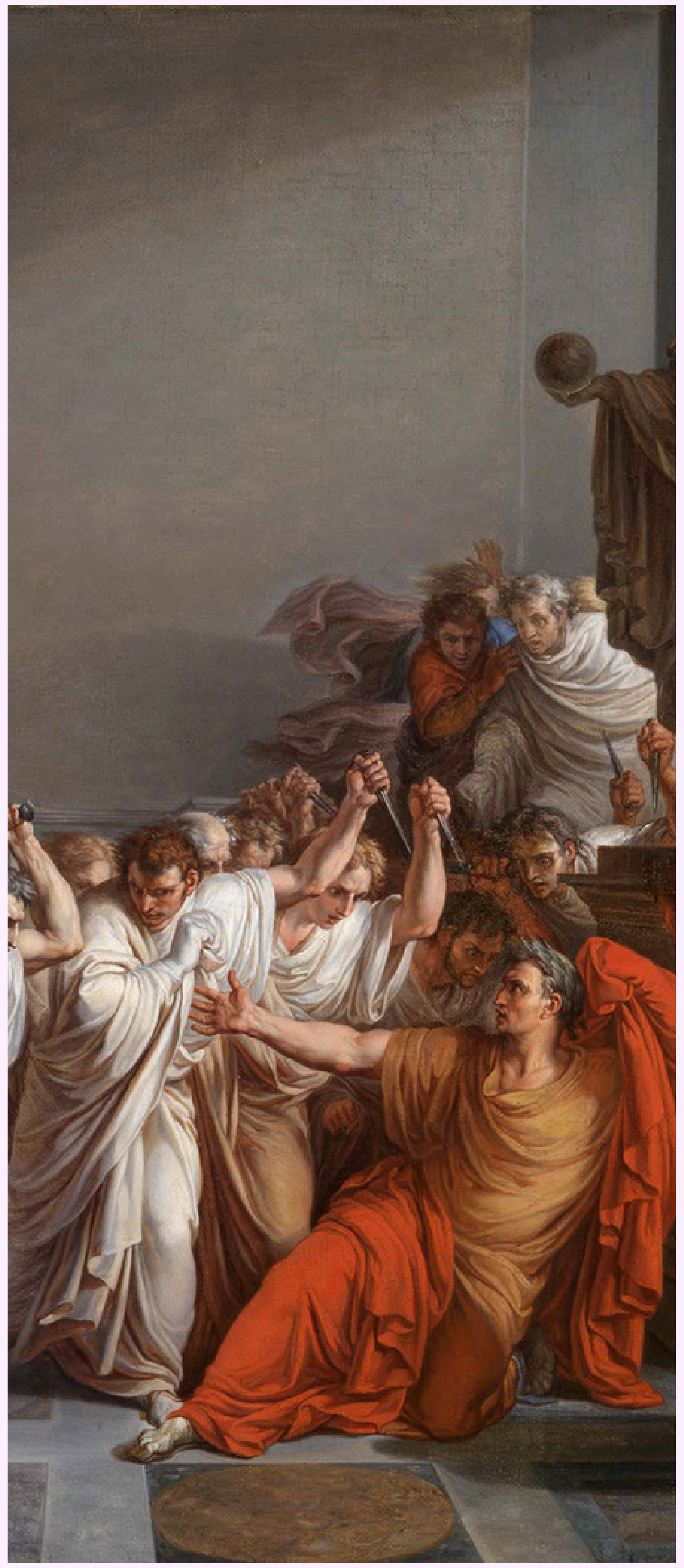
PART 04

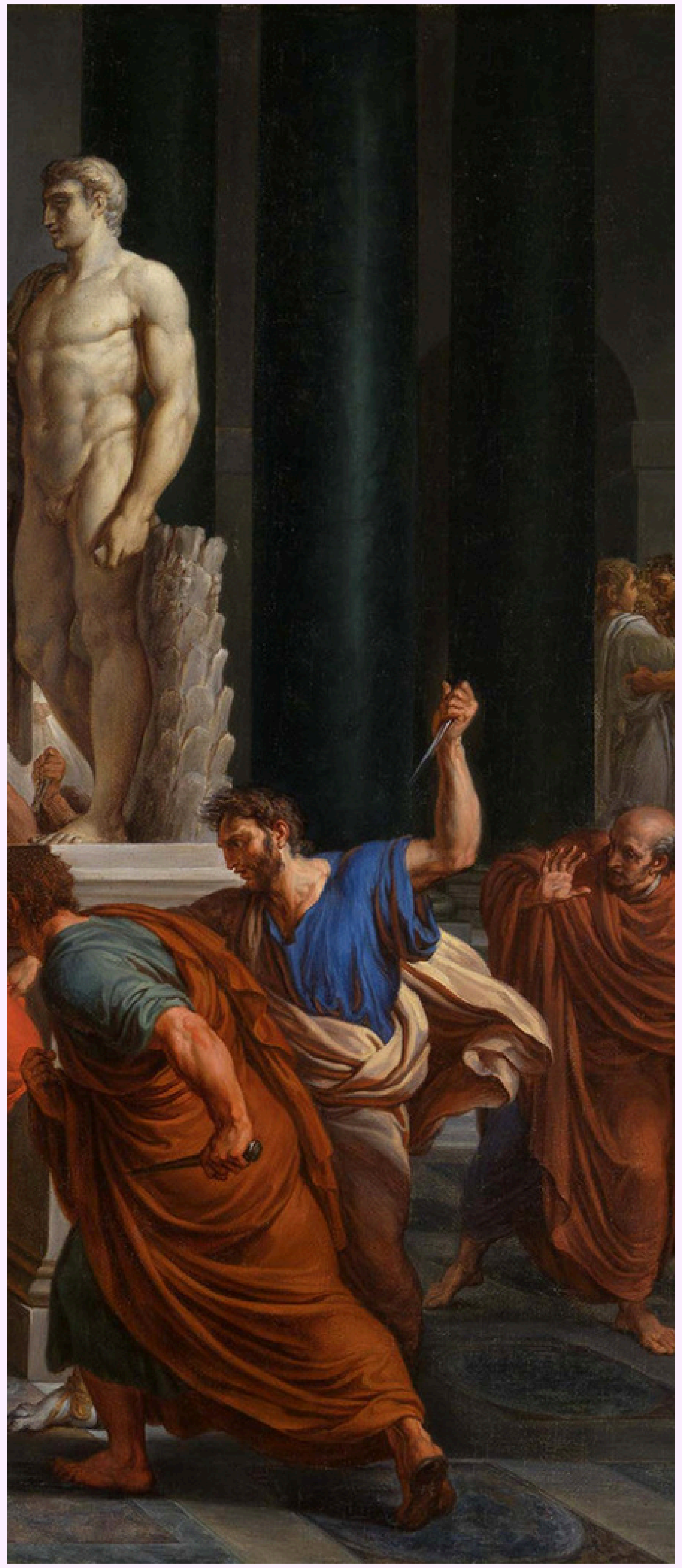
Reasoning about Data

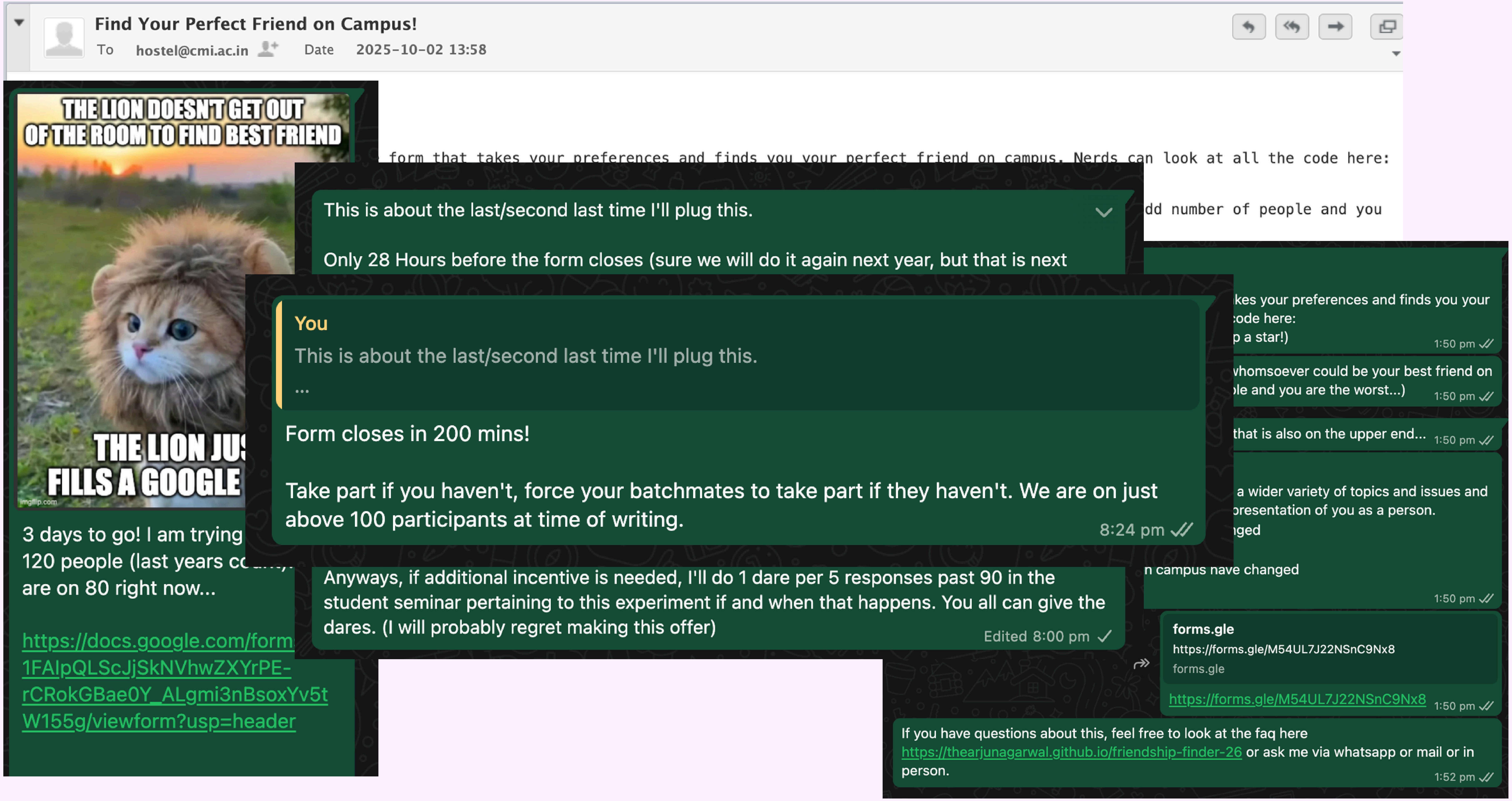
56% people rarely do anything!


33% of Indians are on Social Media


52% of Indians have access to the internet


55% of Indians eat meat


56% of CMI took part in Friendship Finder

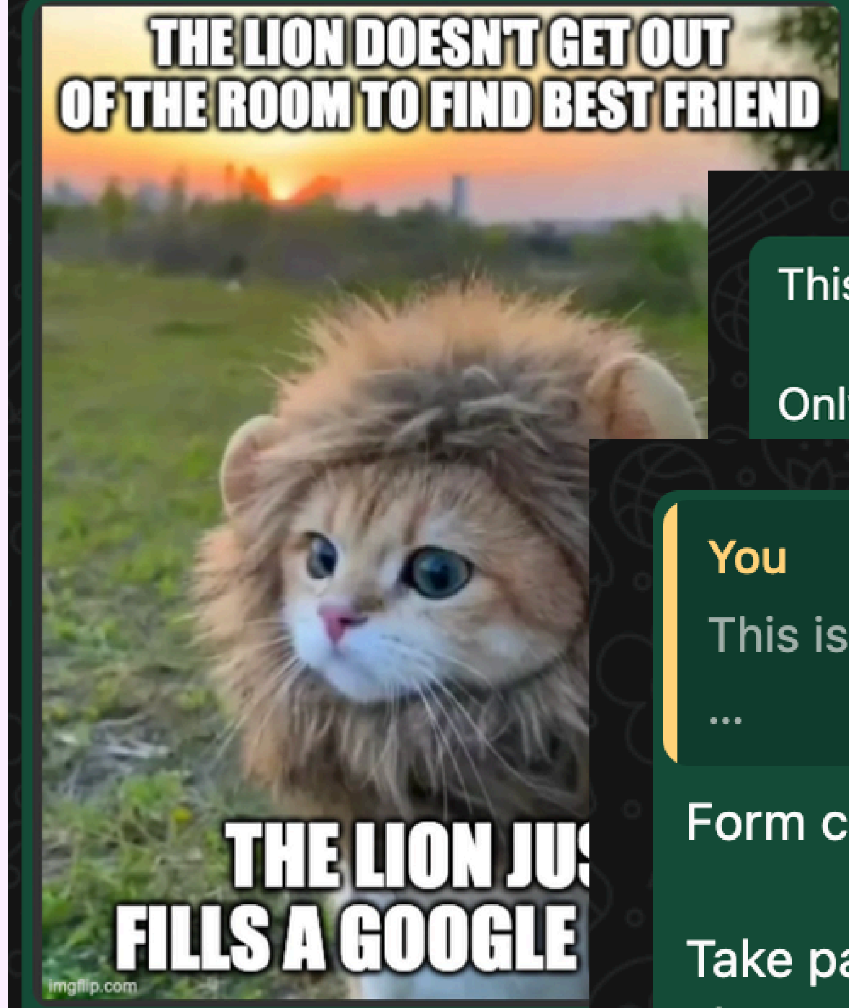


Find Your Perfect Friend on Campus!

To hostel@cmi.ac.in Date 2025-10-02 13:58

form that takes your preferences and finds you your perfect friend on campus. Nerds can look at all the code here:

dd number of people and you



3 days to go! I am trying
120 people (last years co
are on 80 right now...

https://docs.google.com/form 1FAIpQLScJjSkNVhwZXYrPE-rCRokGBae0Y_ALgmi3nBsoxYv5t W155g/viewform?usp=header

This is about the last/second last time I'll plug this.

Only 28 Hours before the form closes (sure we will do it again next year, but that is next

You

This is about the last/second last time I'll plug this.

...

Form closes in 200 mins!

Take part if you haven't, force your batchmates to take part if they haven't. We are on just above 100 participants at time of writing.

8:24 pm

Anyways, if additional incentive is needed, I'll do 1 dare per 5 responses past 90 in the student seminar pertaining to this experiment if and when that happens. You all can give the dares. (I will probably regret making this offer)

Edited 8:00 pm

kes your preferences and finds you your
code here:
p a star!)

1:50 pm

whomsoever could be your best friend on
ple and you are the worst...)

1:50 pm

that is also on the upper end...

1:50 pm

a wider variety of topics and issues and
presentation of you as a person.
nged

n campus have changed

1:50 pm

forms.gle
https://forms.gle/M54UL7J22NSnC9Nx8
forms.gle

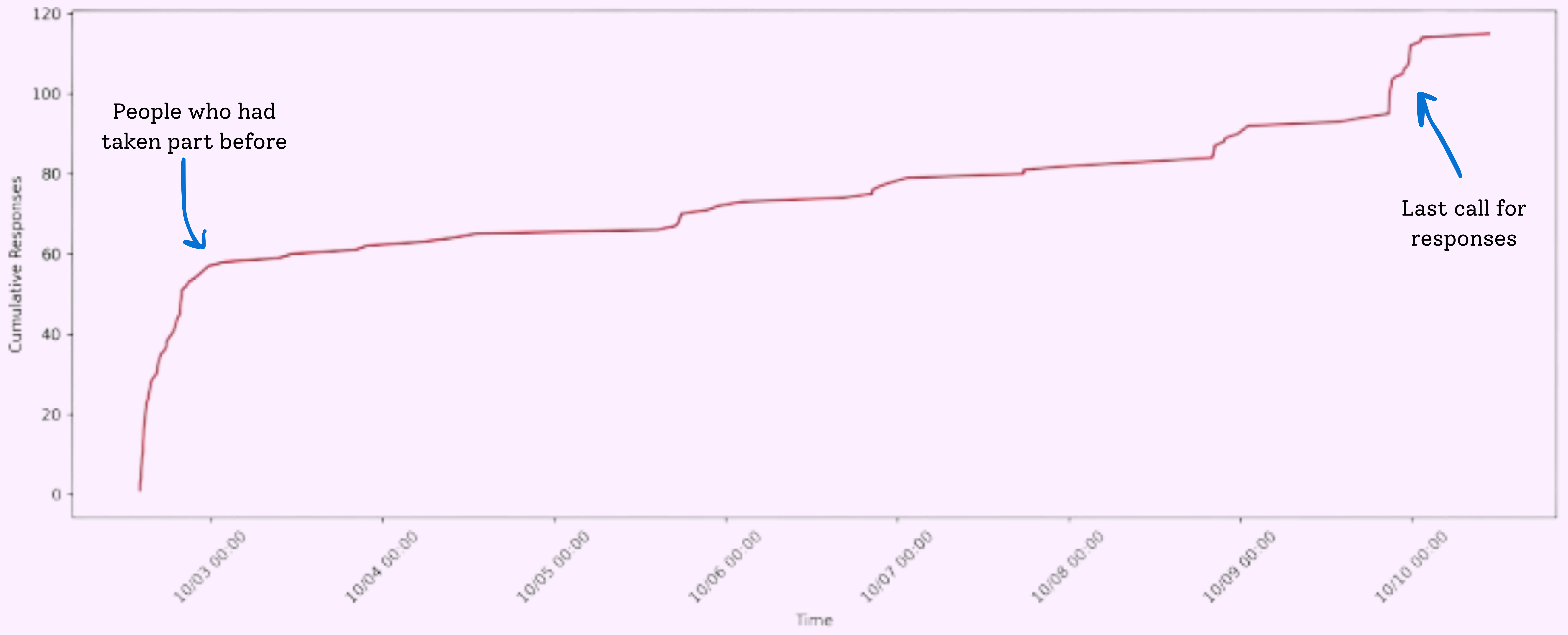
https://forms.gle/M54UL7J22NSnC9Nx8

1:50 pm

If you have questions about this, feel free to look at the faq here https://thearjunagarwal.github.io/friendship-finder-26 or ask me via whatsapp or mail or in person.

1:52 pm

Response Graph



Getting Data

You can have the best matching algorithm, but unless people know about it, it doesn’t matter!

I am not the best marketer, but here is some stuff I had in mind:

- There is a popular maxim in advertising that people will view an ad six times before they pay any attention to it.
- Make it real: People need to believe that this is a real thing. If the questions are stupid or too less, it will seem fake.
- Reduce the friction. No app downloads, no signups, no long responses, a simple privacy policy.
- Make it Fun: At the end of this, we do things to have fun. Convince them that it will be fun!

Data Ethics

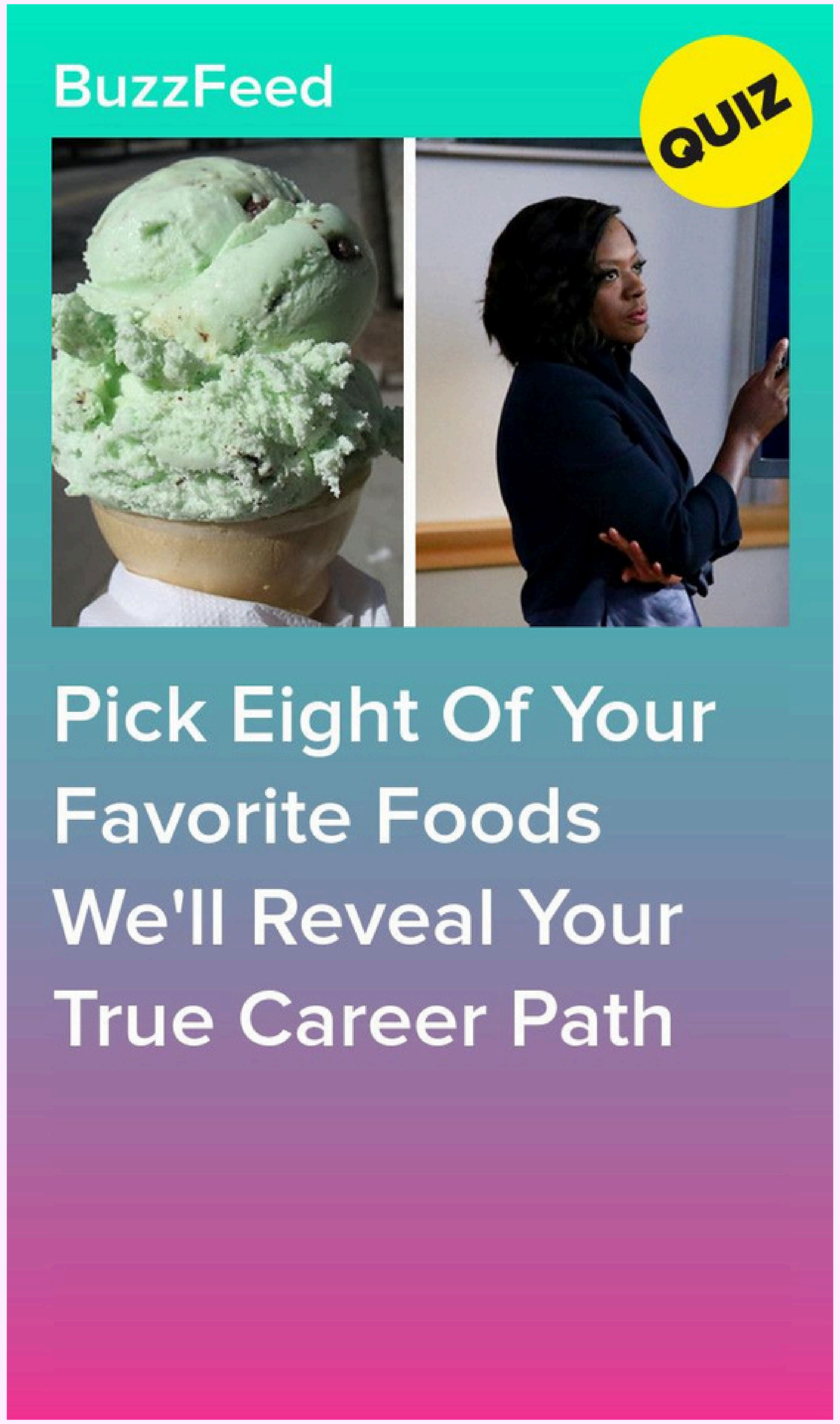
1. Only do with the data what you claim to do.
2. The higher the intervention, the higher the burden of ethics should be.
3. Make policies to make sure you are incentivised to do the right thing.

Your data should serve you!

I take a lot of pride on the fact that the privacy policy is 66 words.

I am not gonna sell your data. I am probably not even gonna look at it. At the end of form time, I will just export the csv which will be the import to my code, which will return csv of matches which will go to a SMTP server (hosted on my computer) and you'll all get the mail. And then I will delete all data.

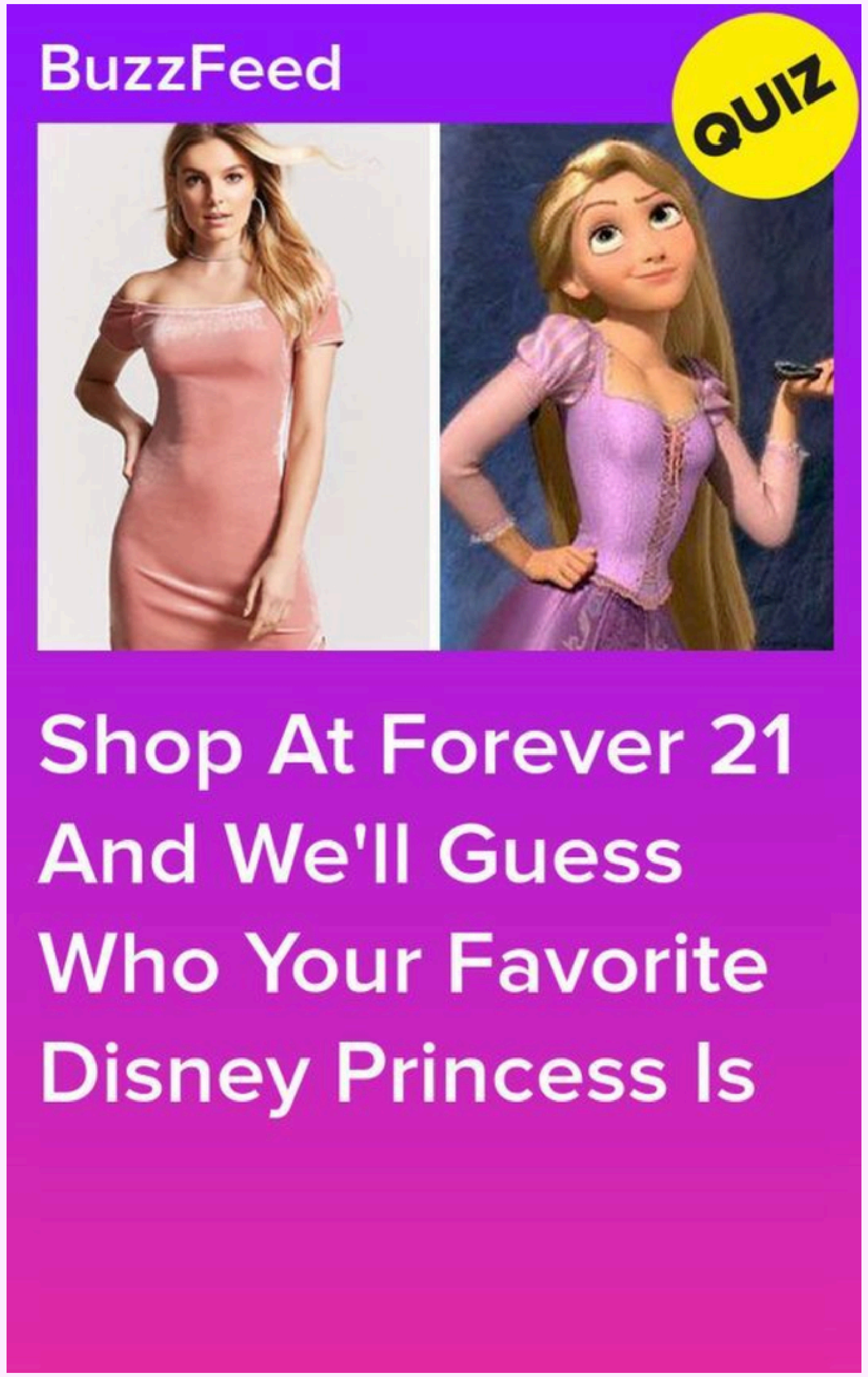
Buzzfeed Tests



BuzzFeed

QUIZ

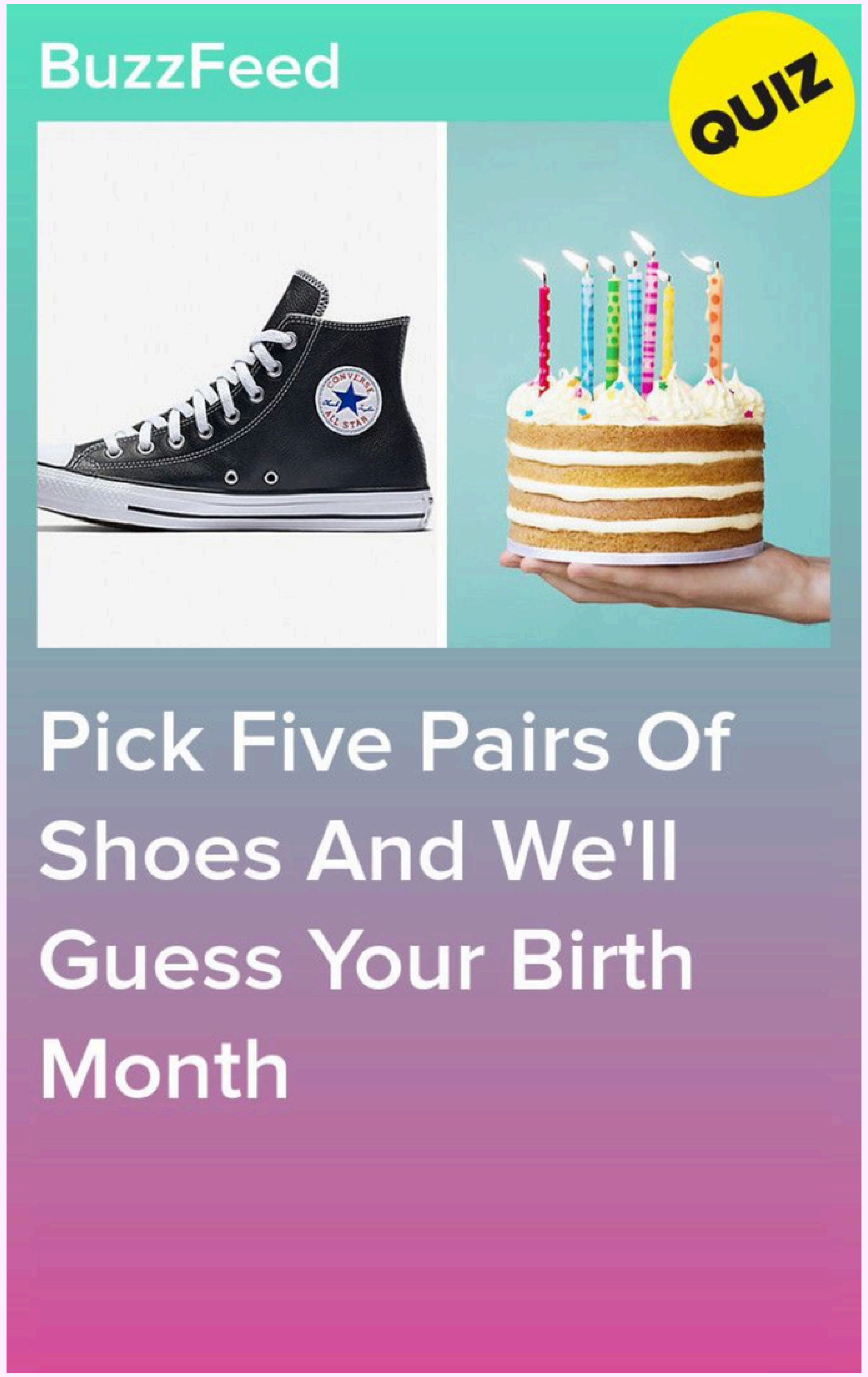
Pick Eight Of Your Favorite Foods We'll Reveal Your True Career Path



BuzzFeed

QUIZ

Shop At Forever 21 And We'll Guess Who Your Favorite Disney Princess Is



BuzzFeed

QUIZ

Pick Five Pairs Of Shoes And We'll Guess Your Birth Month

Modern Love

In the first episode of this series, we get to see Maggie who lives in a building with a doorman.

Whenever she returns from a date, the doorman disapproves of the man; till one a day, he nods. She asks him: why? He replies that it was never about the man. It was about her. If her eyes lighted up when she talked to him, if she seemed genuinely happy.

Instead of asking for a checklist the partner must satisfy, I want to figure out what would light a person up. The answer, someone who is similar enough to be relatable and different enough to have interesting conversations.

Psychometrics

Some people found these quizzes so fun, that they made it their research interest. This is called **Psychometrics.**

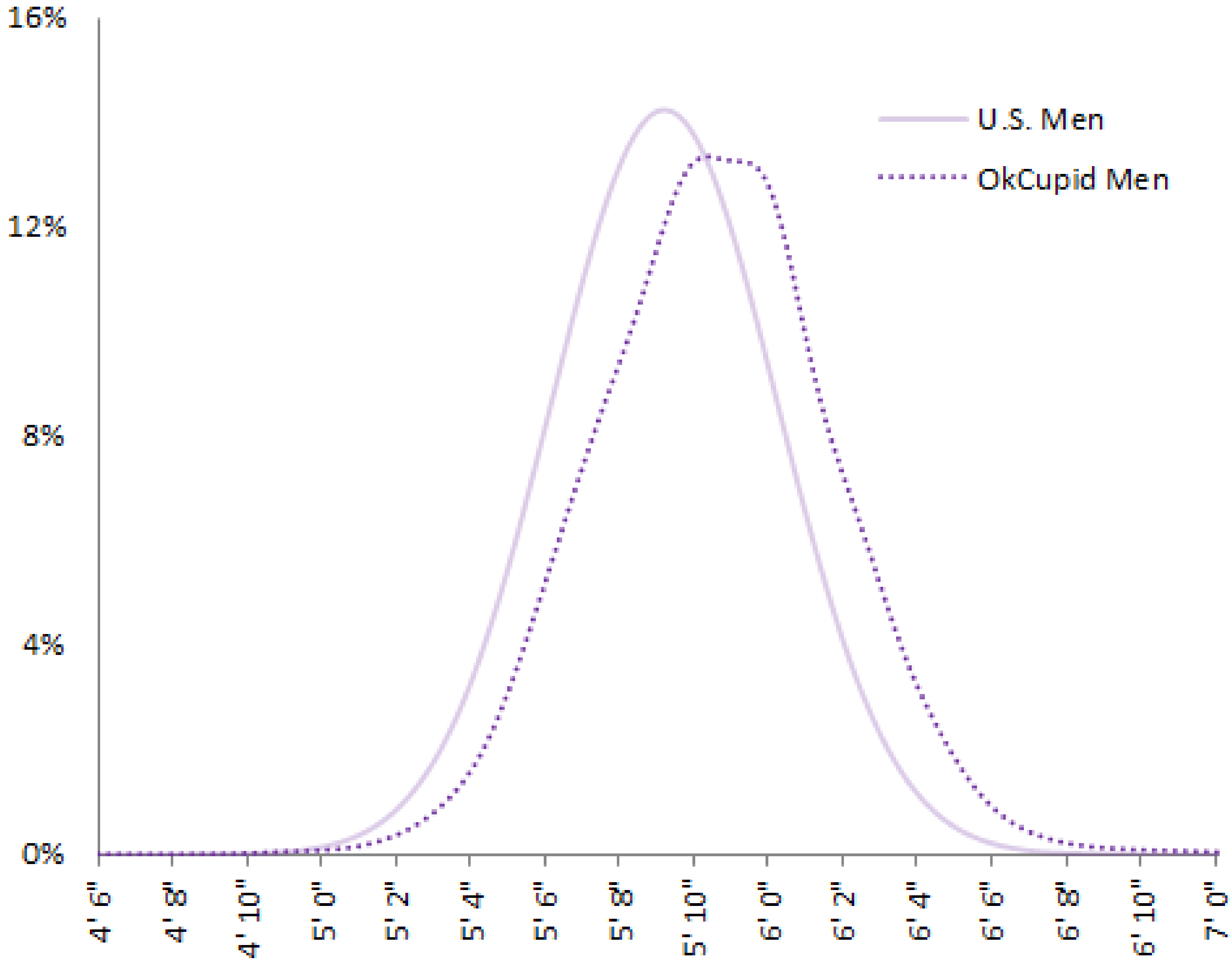
Similar to Stanford Marriage Pact, we ask you to tell us about yourself and use that to compare you to people. Two immidiete choices to make were:

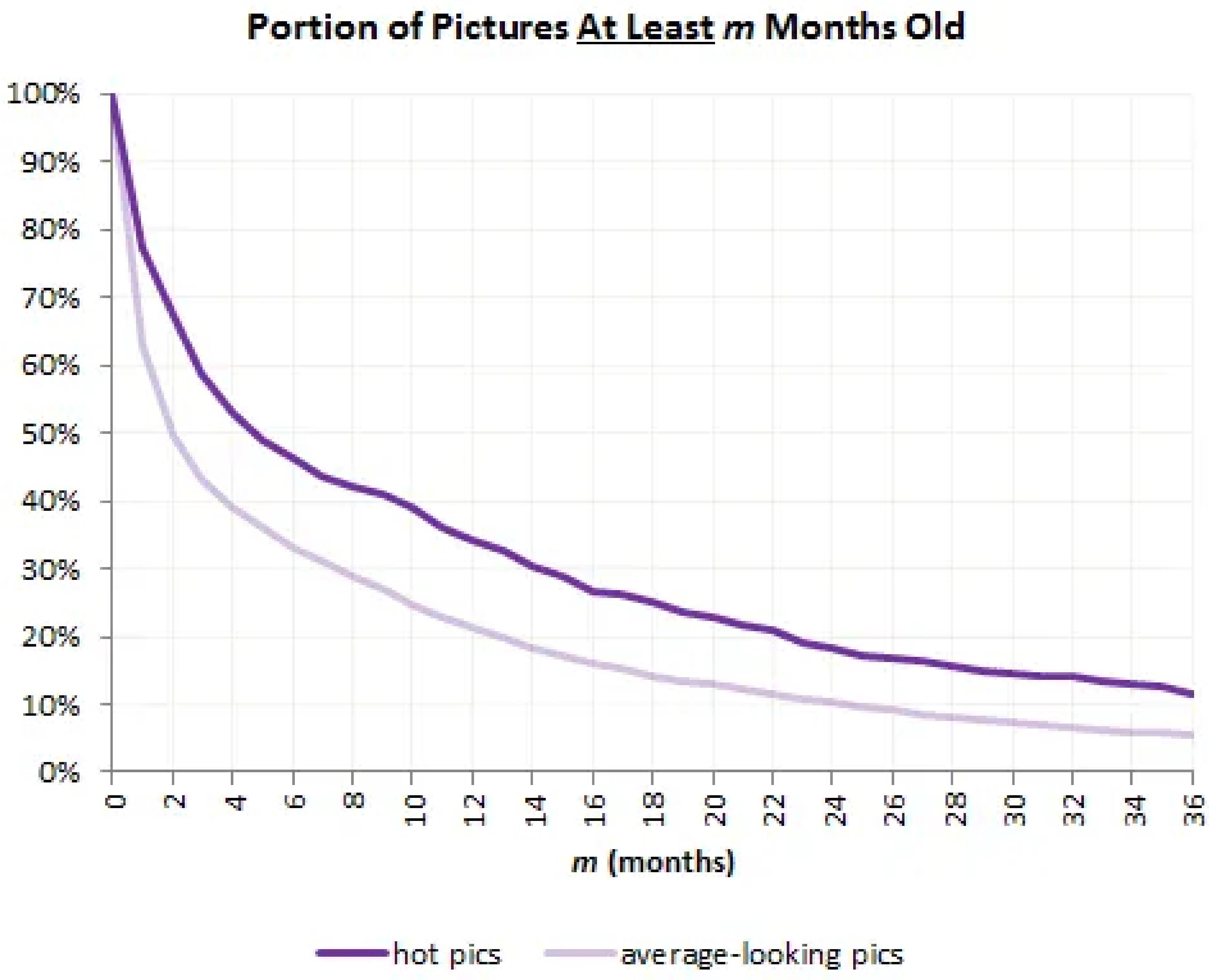
- How many questions to ask?
- How many options to give?

We have to balance between information and friction. Furthermore, people are not perfect evaluators, so we don’t get too much information beyond a point. I choose to use a 10 point scale as it felt natural (and makes a lot of calculations more easy to reason about) but as far as I know any odd number, especially $2^n + 1$ are often good choices.

Want to know who else figured this out? Buzzfeed!

Male Height Distribution On OkCupid





Portion of Pictures At Least m Months Old
100%
90%
80%
70%
60%
50%
40%
30%
20%
10%
0%
0
2
4
6
8
10
12
14
16
18
20
22
24
26
28
30
32
34
36
m (months)
hot pics
average-looking pics

Asking the Right Questions!

If we asked, “I am funny” or “my friends think I am funny”; I guarantee we’ll get a lopsided distribution. Because “I agree” to a rational actor reads like “I am funny and please match me with funny people!”

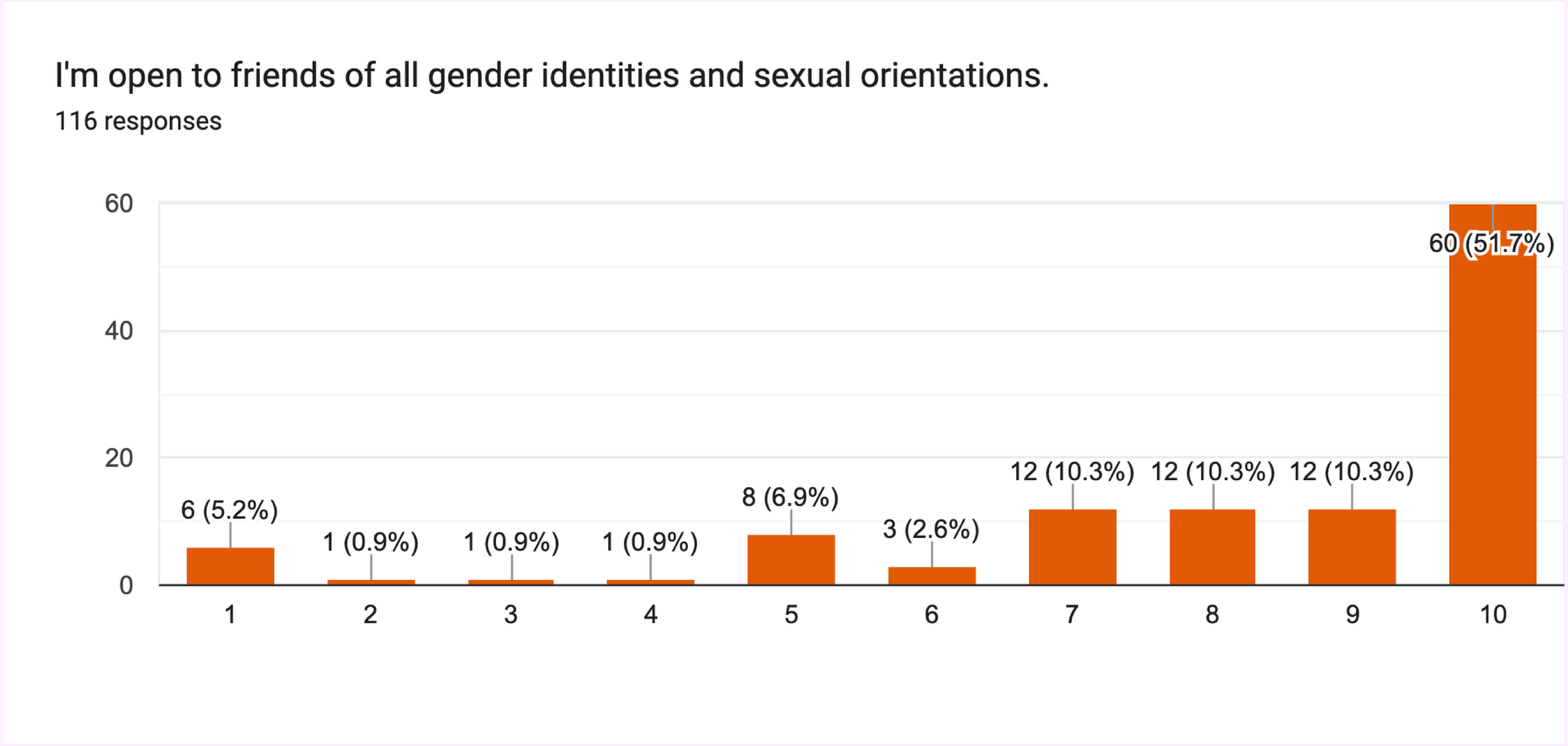
So I have to ask questions like: “Social activism and making a difference are important to me.” cause lying on it makes no sense to a person on the other side of the spectrum.

or more benign: “I’m a night owl.”

or “I spend significant time on internet forums or online communities”

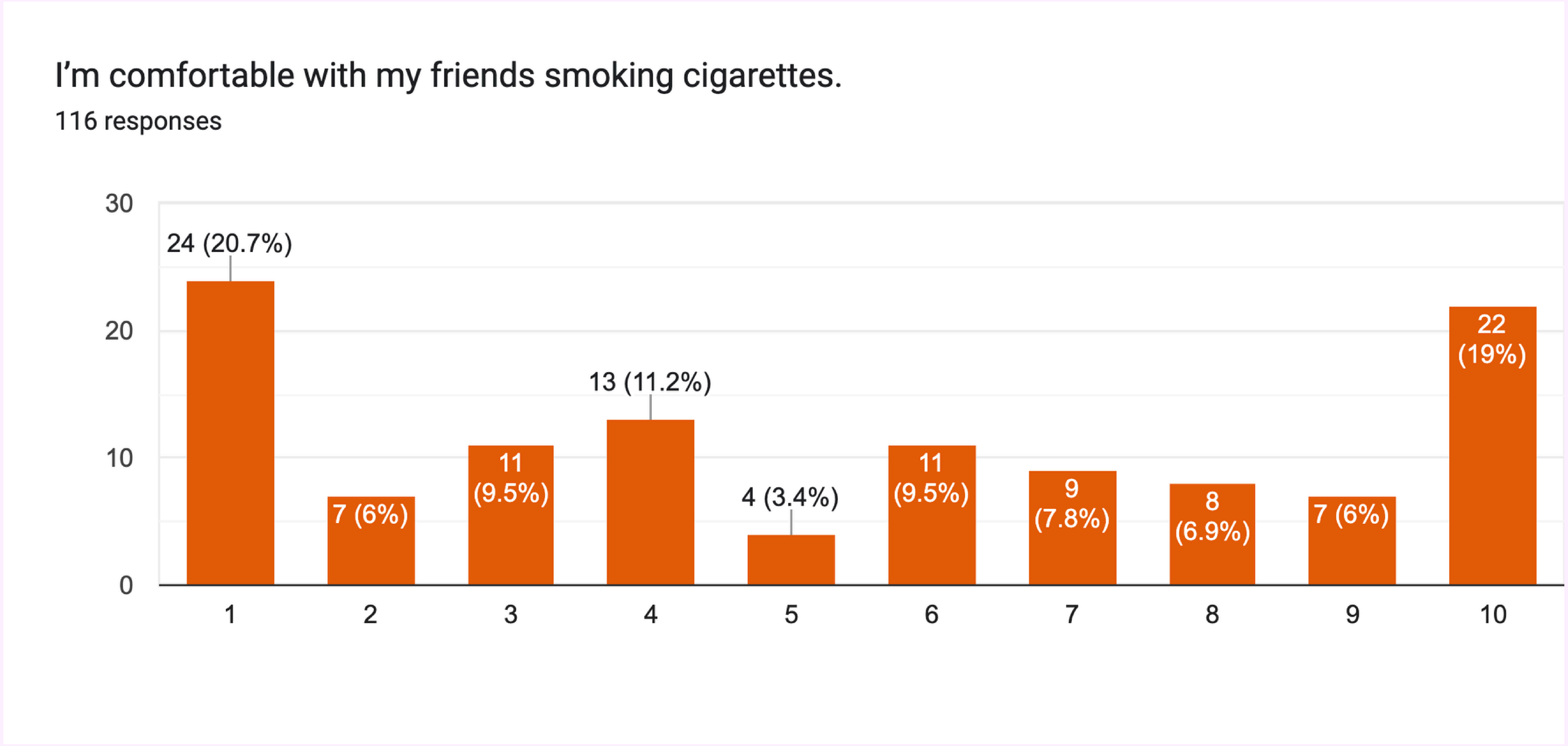
The idea is that a questions should be such that the respondent should not have incentive to misrepresent themselves.

How much information is it anyway?


I'm open to friends of all gender identities and sexual orientations.

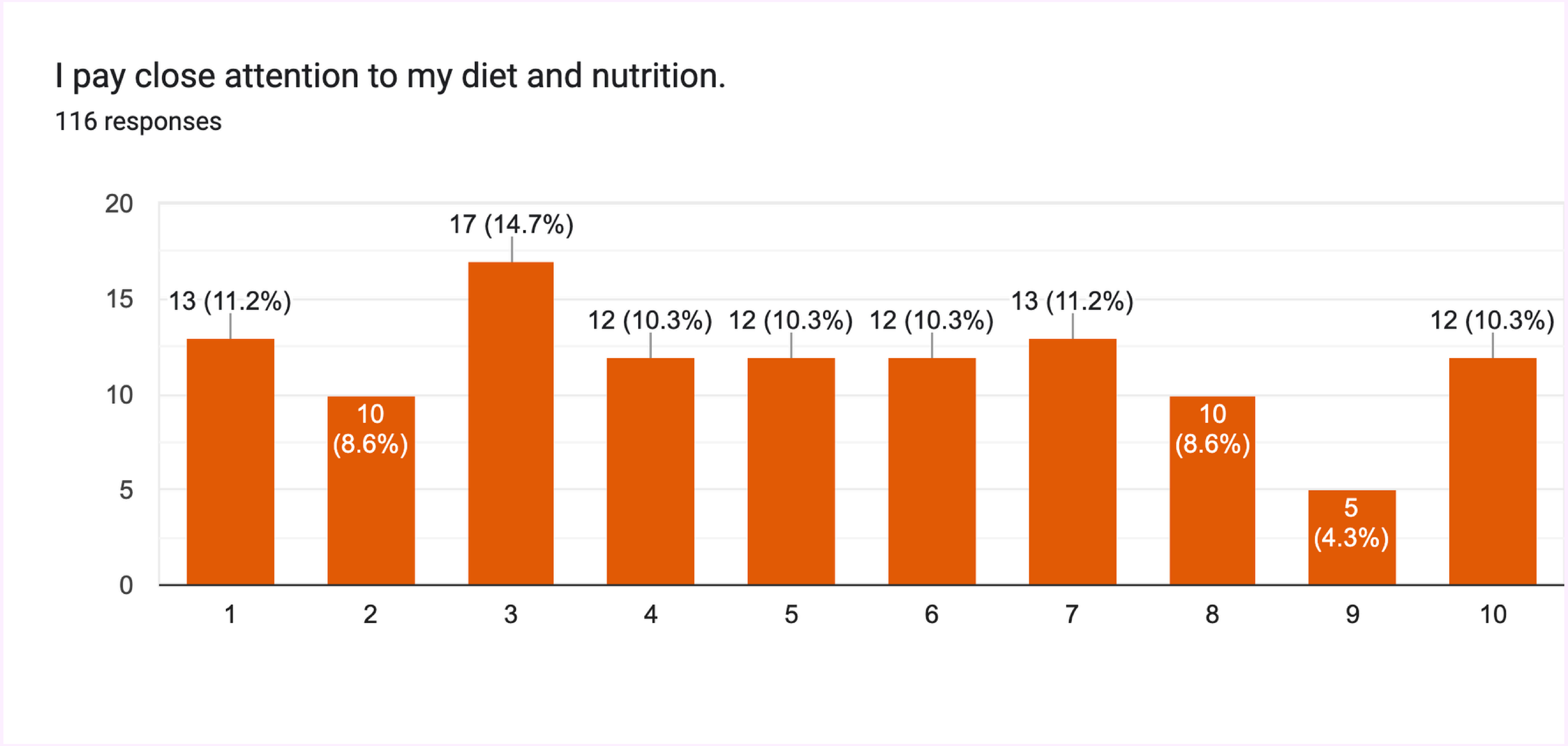
116 responses

How much information is it anyway?

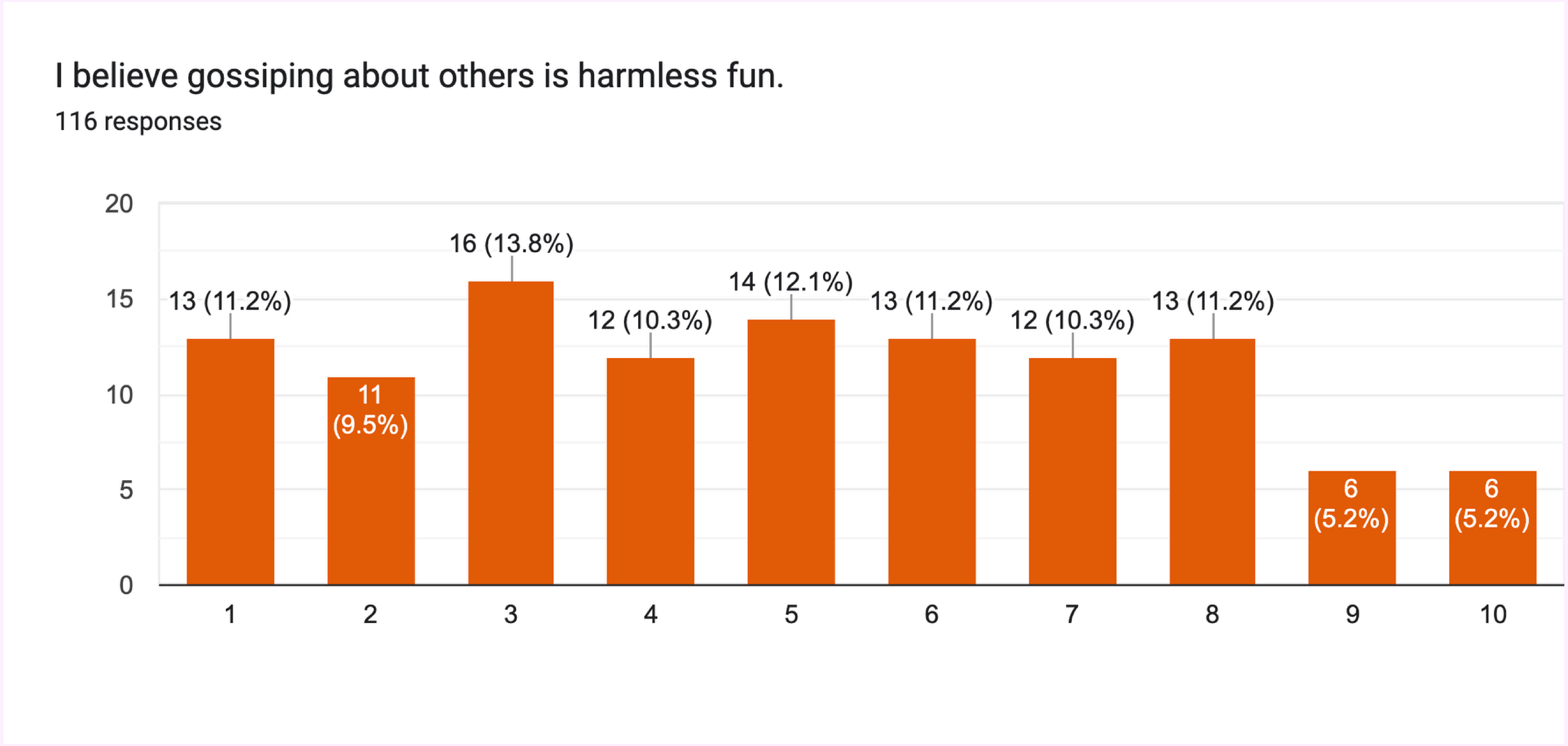

I'm comfortable with my friends smoking cigarettes.

116 responses

How much information is it anyway?



How much information is it anyway?



I believe gossiping about others is harmless fun.

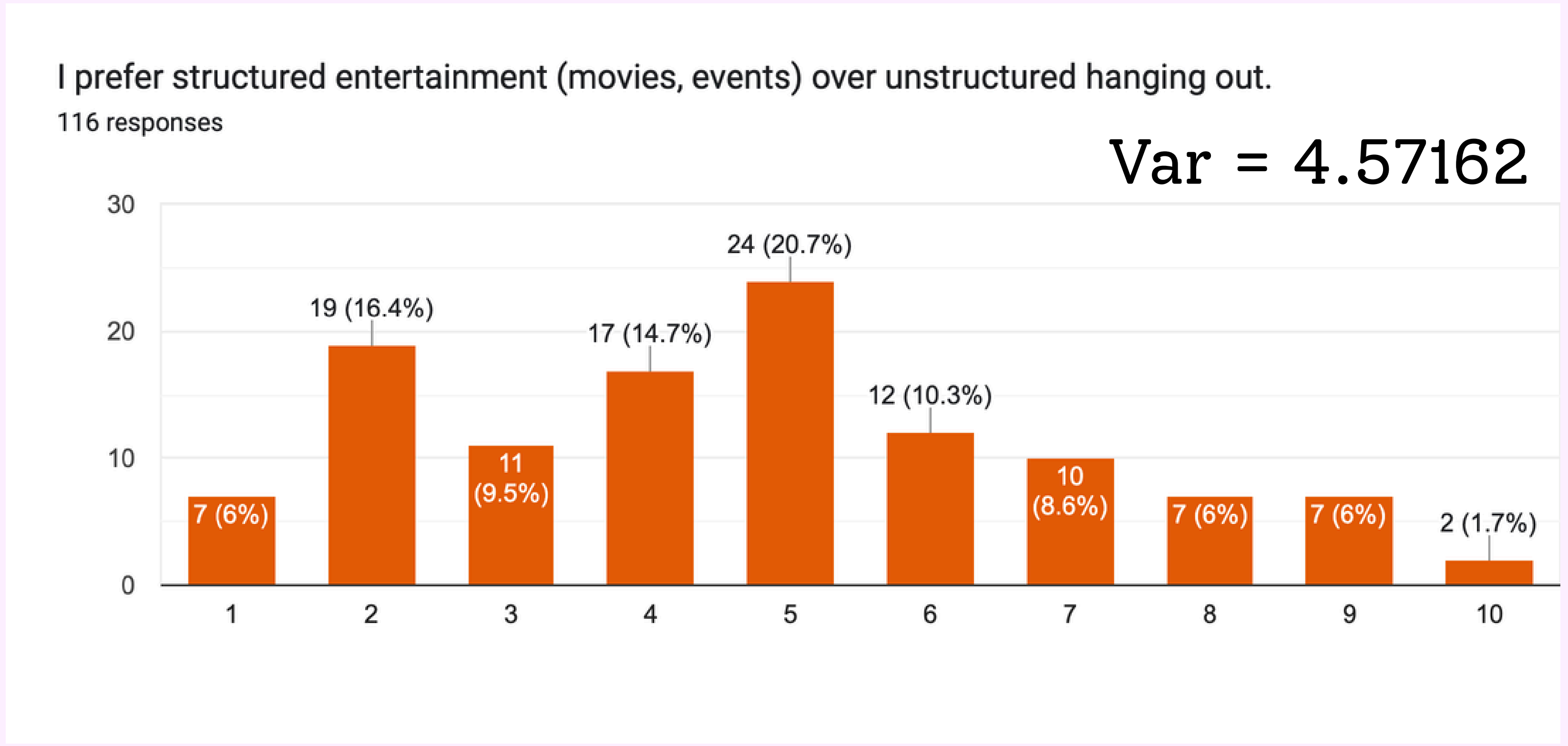
116 responses

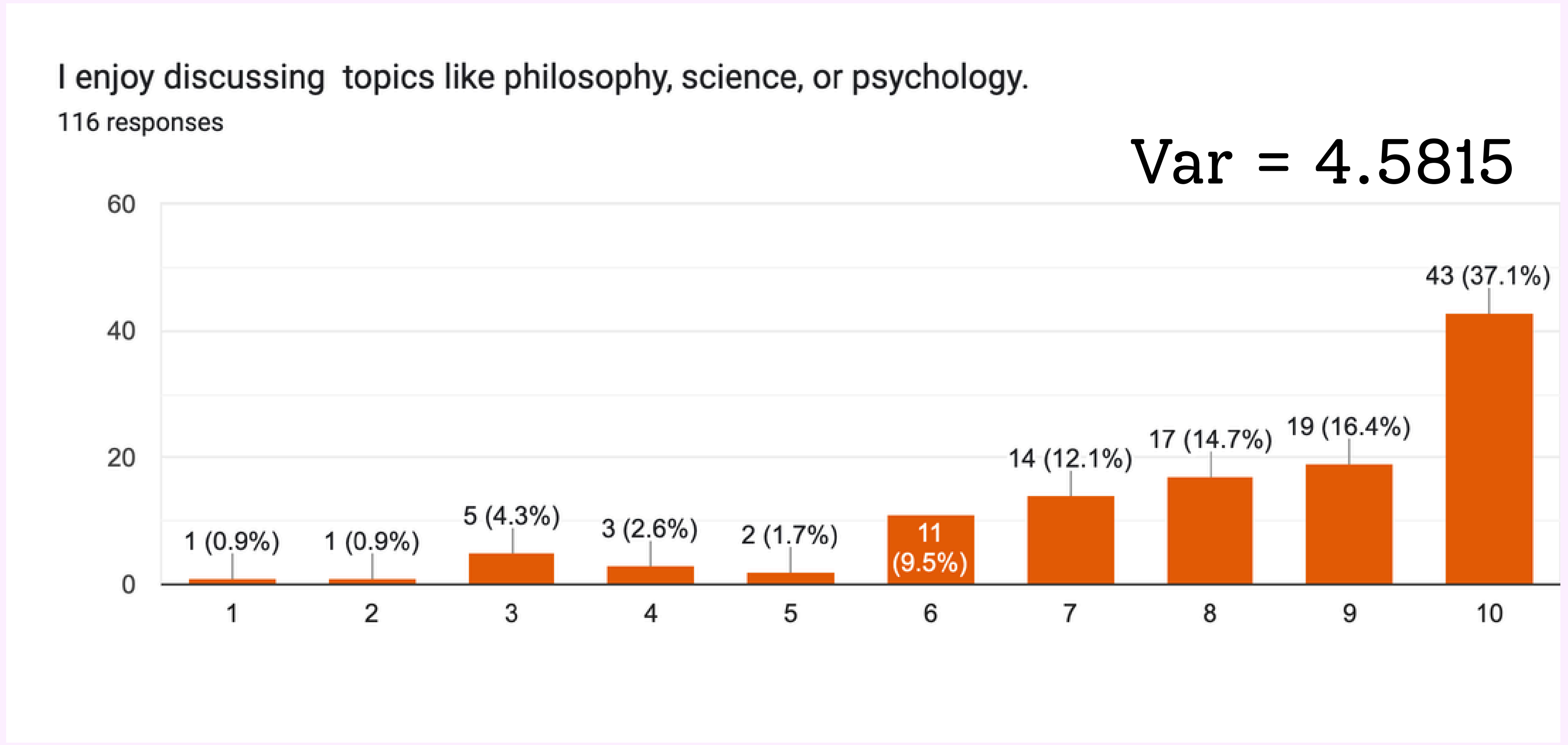
How much information is it anyway?

A very lopsided distribution doesn't tell us much and uses up a question. Ideally, we want questions to have almost even distributions. Although, some questions are important enough that we let go of the information and just ask them.

Is there a way to measure this?

Varience doesn't work as it is low for data which is lopsided towards the center, while high for even distriubutions.



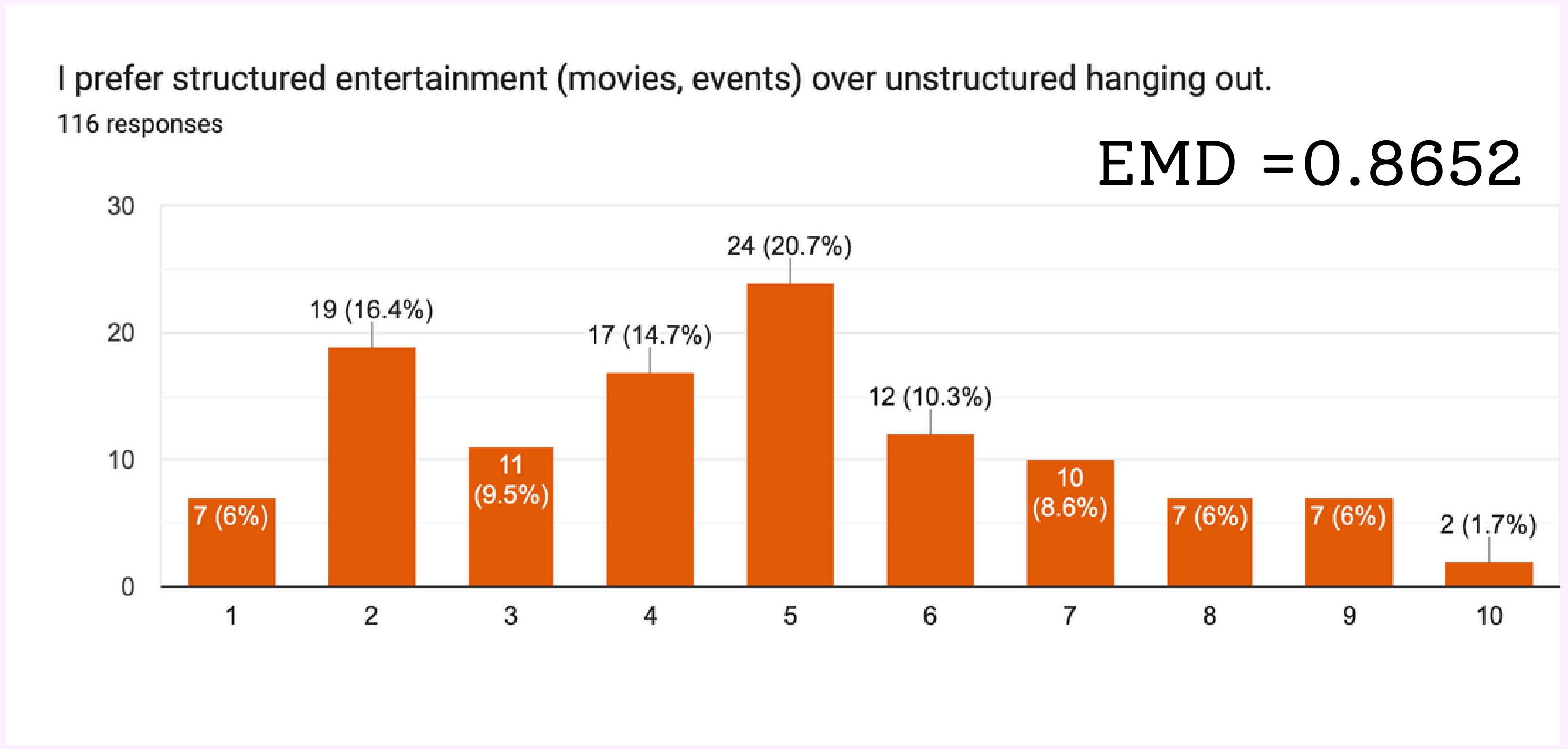


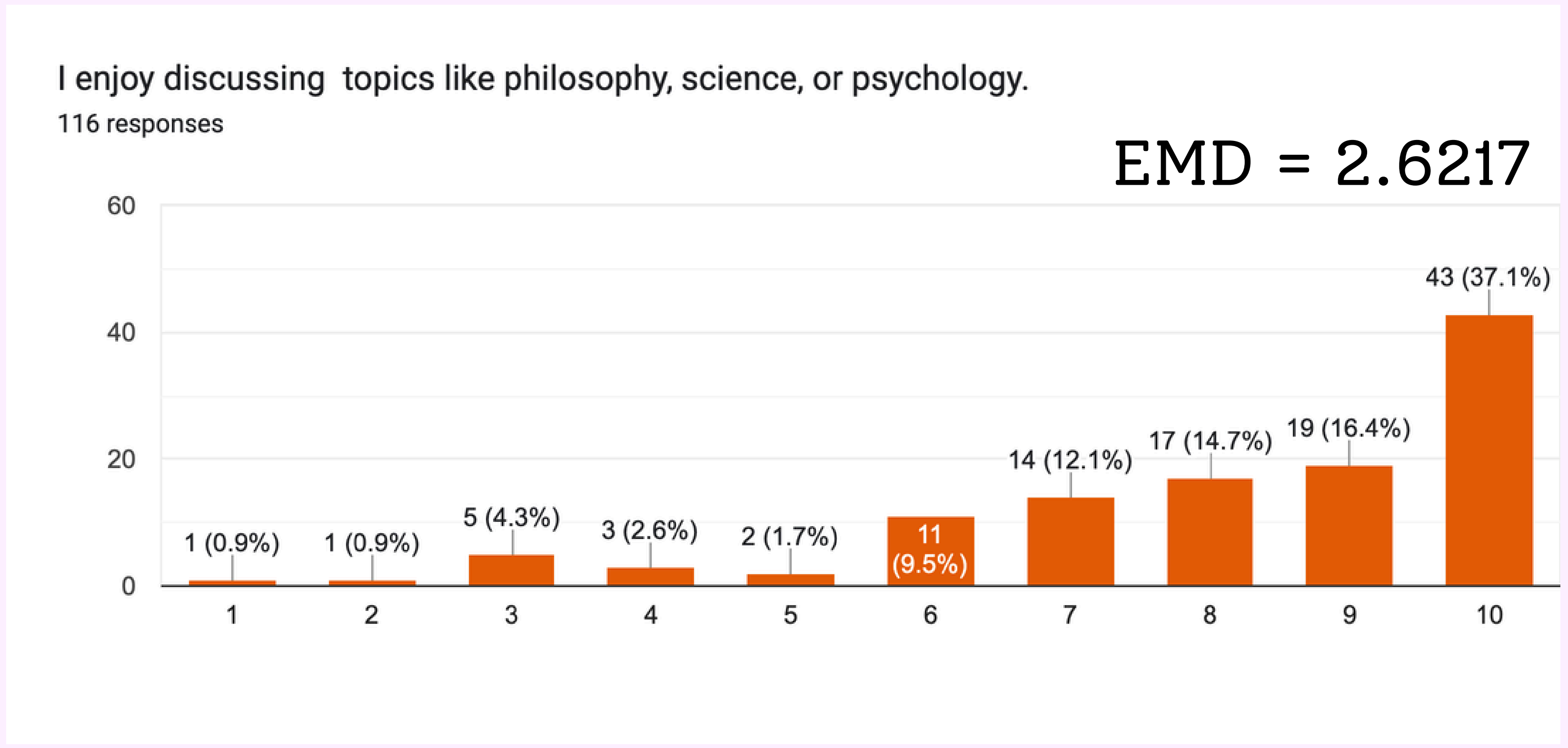
Earth Mover's Metric

A method of evaluating this, suggested to me by Prof. Rajeeva Karandikar was Wasserstein metric or Earth Mover's Distance.

The idea is, consider the bar graph we have to be a pile of earth. How much 'movement' do we need to do to make the pile even? We define movement to be the number of points we need to move times the average distance.

This can be computed by a very nice recursion in O(n) time (where n is the number of options).

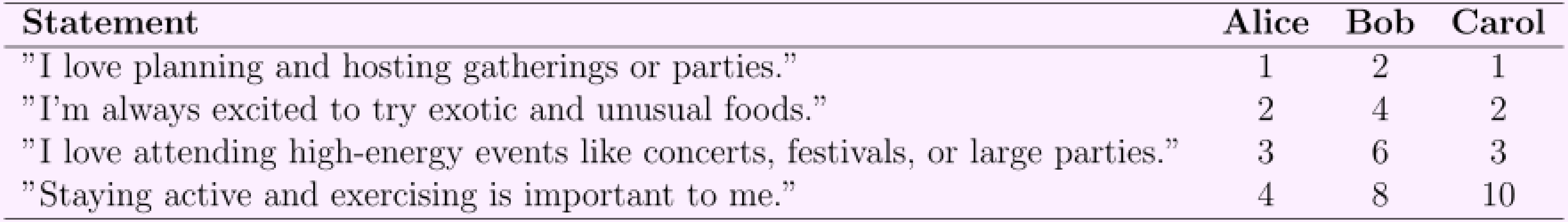




PART 05

The Quality of Matches

Choosing a Metric



Statement	Alice	Bob	Carol
”I love planning and hosting gatherings or parties.”	1	2	1
”I’m always excited to try exotic and unusual foods.”	2	4	2
”I love attending high-energy events like concerts, festivals, or large parties.”	3	6	3
”Staying active and exercising is important to me.”	4	8	10

Who are the closest?

Bob and Carol: Are closest if we take the Euclidian distance
Alice and Bob: Are the same people, Bob just is twice Alice in everything.
Alice and Carol: Differ the least in total

Which answer is correct? As it turns out, all of them.

Choosing a Metric

Given all of us are points in a 64 dimensional space, how do we define the distance between people? We define a metric (or a divergence in this case as the triangle inequality is not satisfied).

The problem is that every metric imposes some idea of what two people being close means.

Additionally, We have an additional requirement that computing the distance between two points should be fast.

The problem is that this doesn’t narrow our options.

Jigsaw

Danial Sloss's comedy special Jigsaw is (in)famous for causing half a million breakups and more than 1000 divorces world-wide. (Daniel stopped counting in 2022). I believe it an excellent analogy for what I believe love and friendship are.

While I'll not spoil much of this excellent piece (as I recommend you watch it, possibly with your significant other).

We are all trying to assemble the jigsaws of our lives, and some pieces are extra and some missing. The problem is, we can't treat someone else as extra pieces. They are their own complex person. The only case the pieces will fit, is if your jigsaws are similar/compatible enough but different enough to have different pieces.

People are more in love with the idea
of love than the person they are with.

The Metric I used!

This lead to the decision to use a linear combination of mean absolute difference and pearson correlation.

The biggest issue with Pearson in this case is that {1,2,3,4} and {2,4,6,8} are close despite being clearly different people.

The biggest issue with Mean Absolute Difference is that from the pov of {1,2,3,4}, {1,2,3,10} is closer than {3,4,5,6}.

We can fix both these issues by combining the metrics. This will lead to people who have similer pattern of responses and are close to each other.

The way I choose the ratio was by back testing (running it on the data set and seeing if the matches made sense). This is perhaps the most fiddly part of this. The biggest issue is quantifying the quality of matches without a metric as we are choosing a metric. I have experimented way, way too much with it and I don’t have any better ideas than just moving the numbers around and seeing if it is ‘better’.

06

PART 06

Findings

Political Survey?

We were able to see a negative correlation ($\tau = -0.429$, $p < 0.0001$) between political acceptance ("I would mind if my friend had different political views") and dislike of billionaires("I believe billionaires should not exist.").

We also were able to see a positive correlation in people who are comfortable with drinking, smoking and drugs. ($\tau = 0.497, 0.447, 0.398$; $p < 0.0001$ for all).

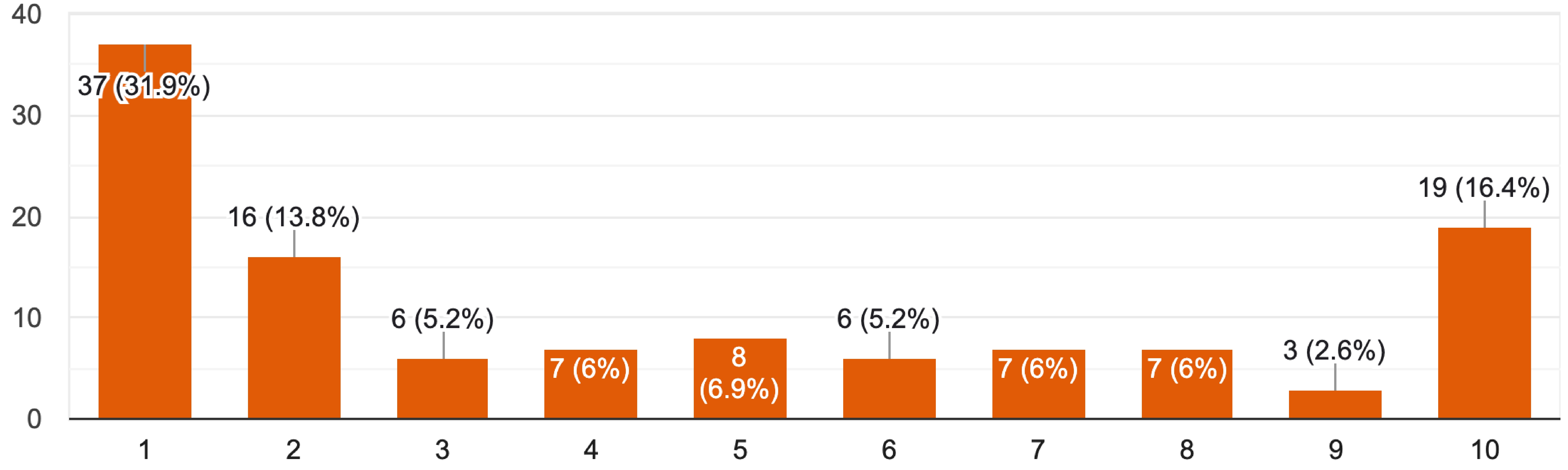
Party hosts are also normally societally conscious. ("I love planning and hosting gatherings or parties" <-> "Social activism and making a difference are important to me.": $\tau = 0.324$, $p = <0.0001$).

Finally, people comfortable with drinking are more likely to be LGBTQIA+ friendly than thoose comfortable with drugs or cigarettes($\tau = 0.322, 0.22, 0.23$).

We got a lot more as well, I'll put them up as well but these were the ones I found the most interesting mainly as they revealed some facts about the political landscape we are in and how it can blend into a "for fun" test.

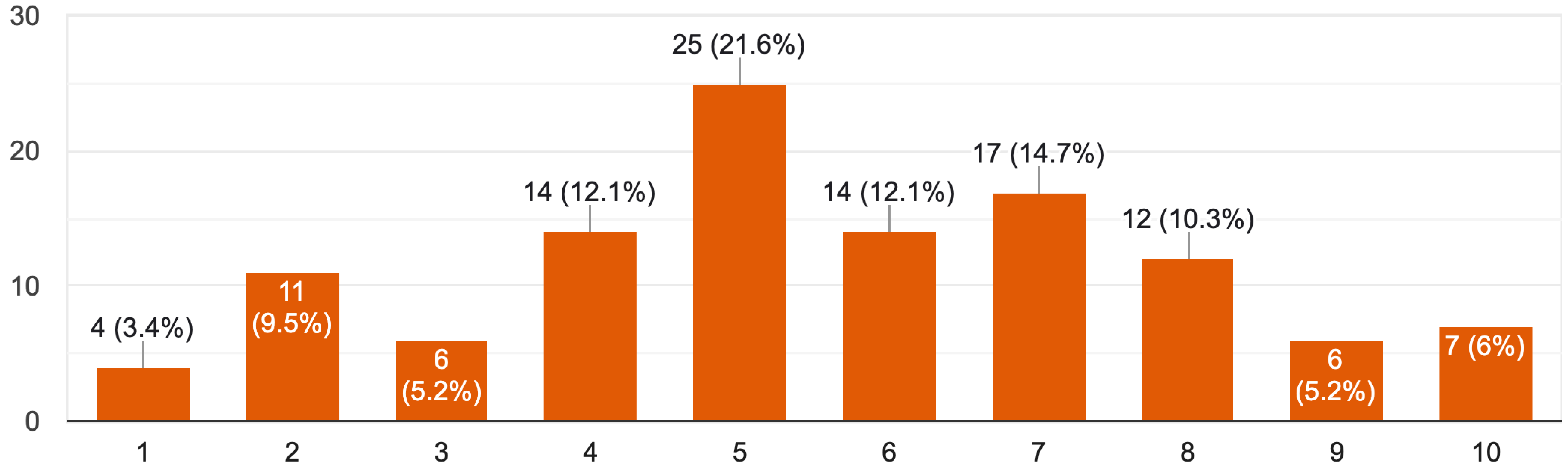
Following a specific sport or team (e.g., cricket, football, F1) is a major hobby for me.

116 responses



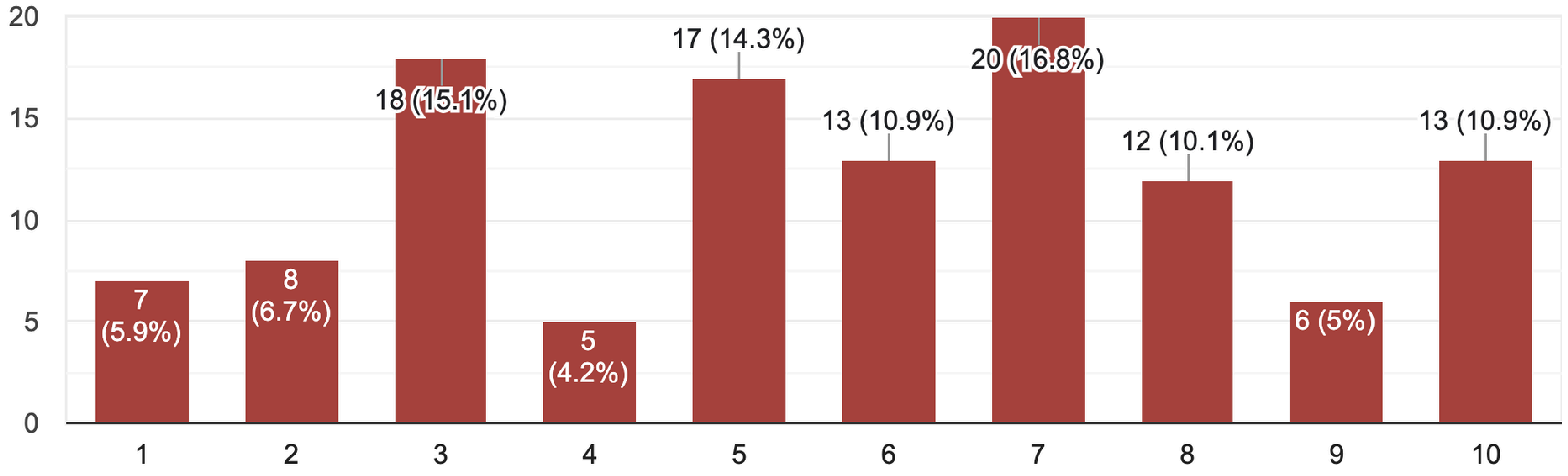
Financial stability is more important to me than chasing passion projects.

116 responses



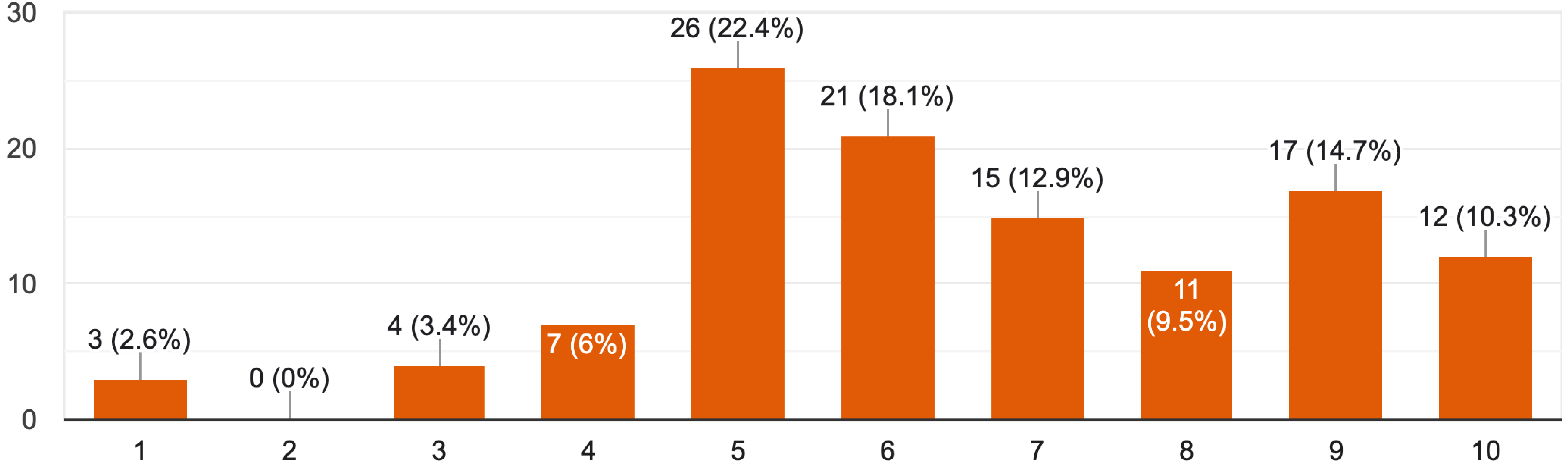
Financial stability is more important to me than chasing passion projects.

119 responses



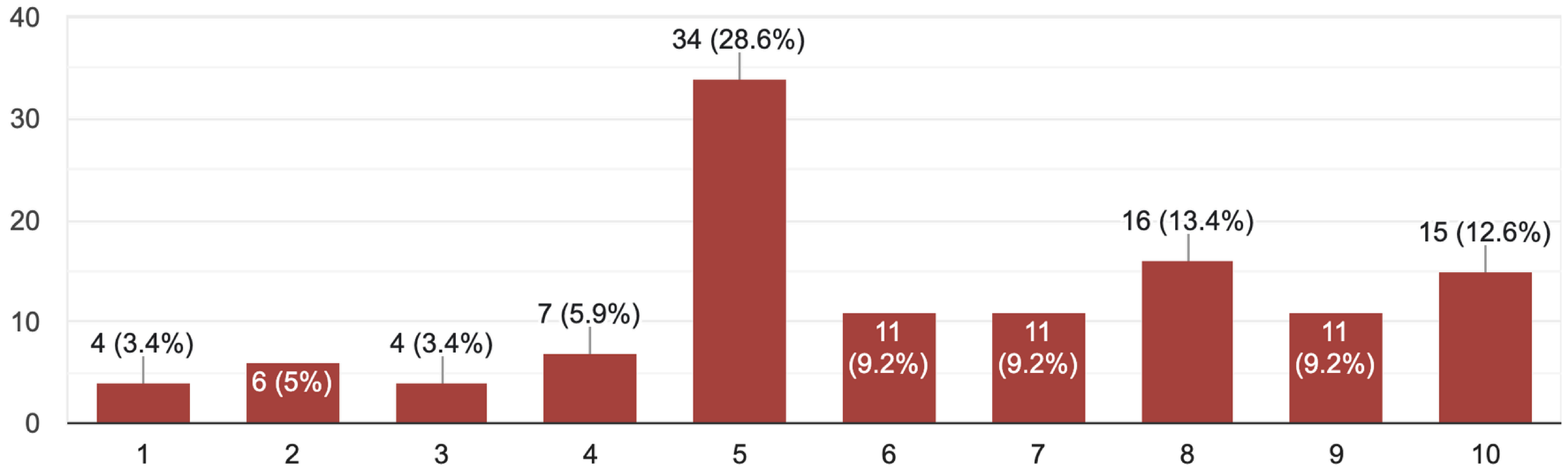
I feel one should choose to buy from local businesses over large corporations.

116 responses

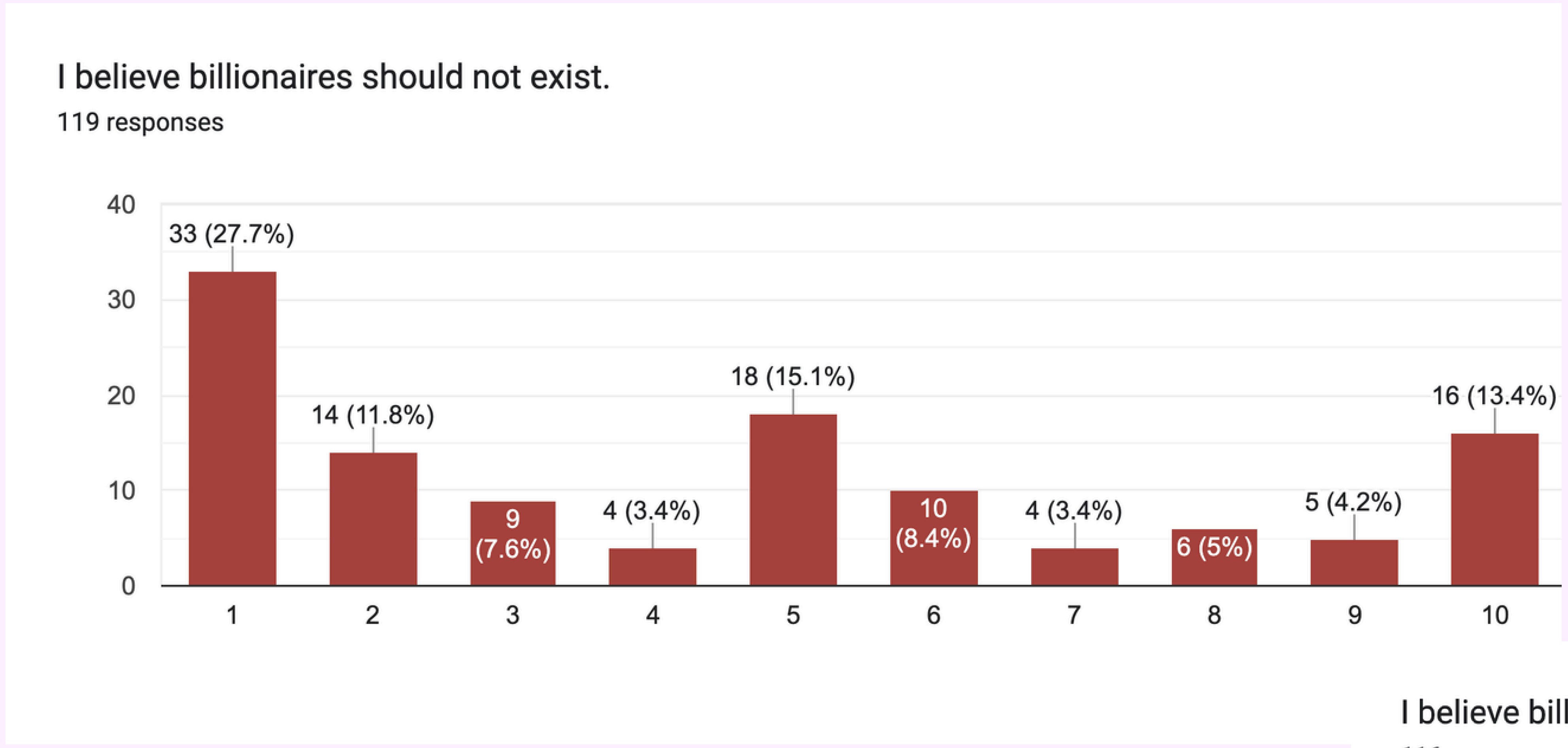


I feel one should choose to buy from local businesses over large corporations.

119 responses



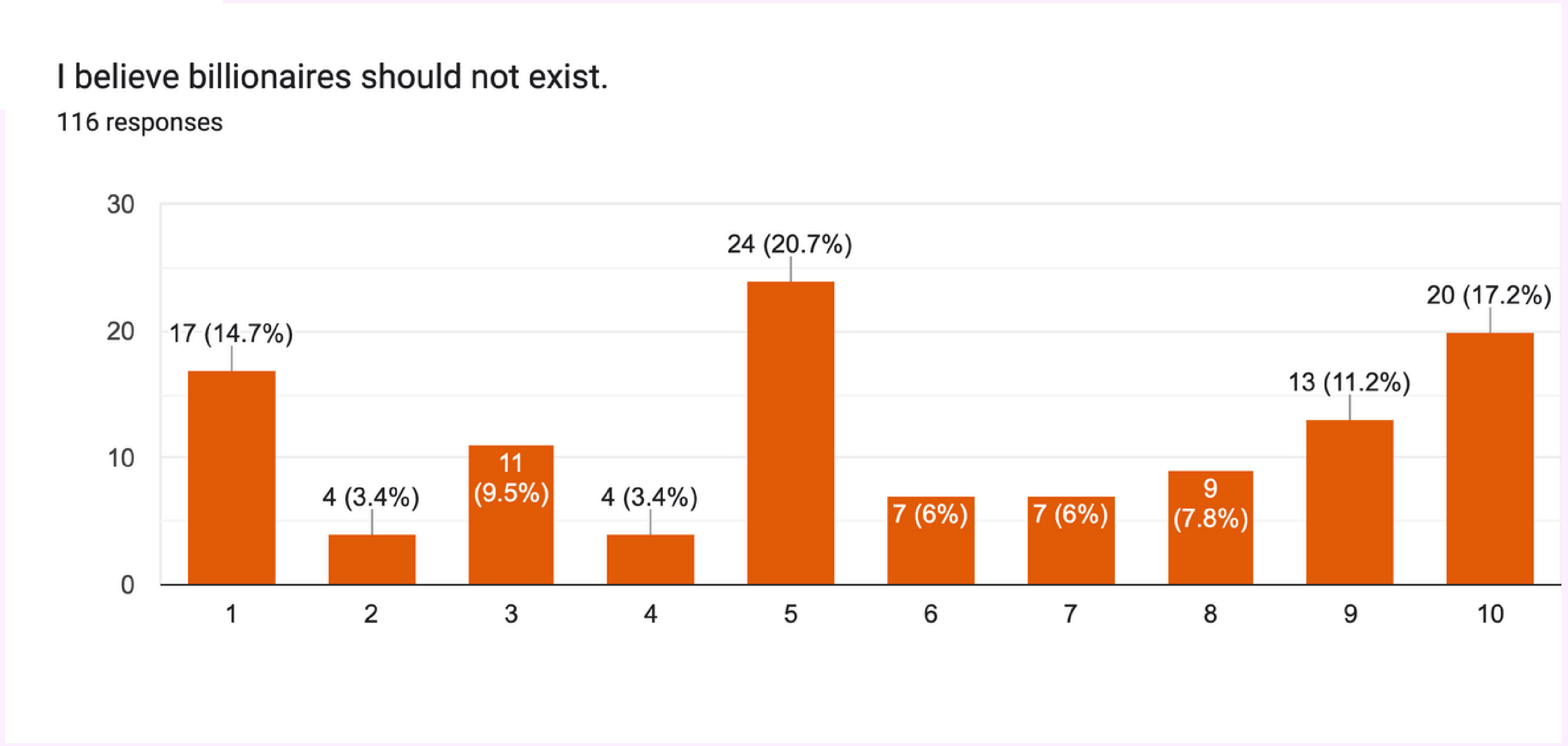
Political Survey??



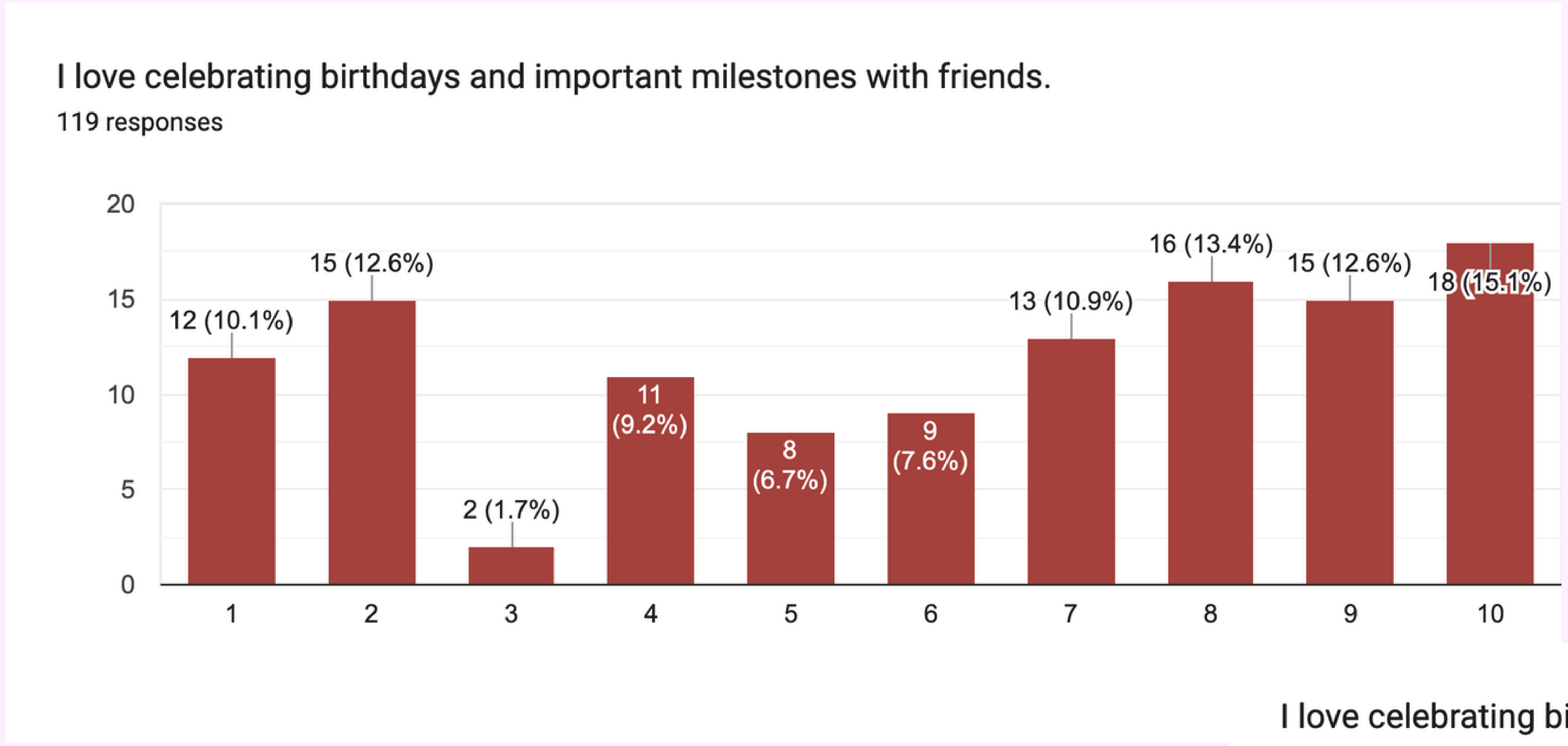
People have turned against at a moderate rate in a significant fashion.

d=0.4234597169528678

p = 0.001368346746250703

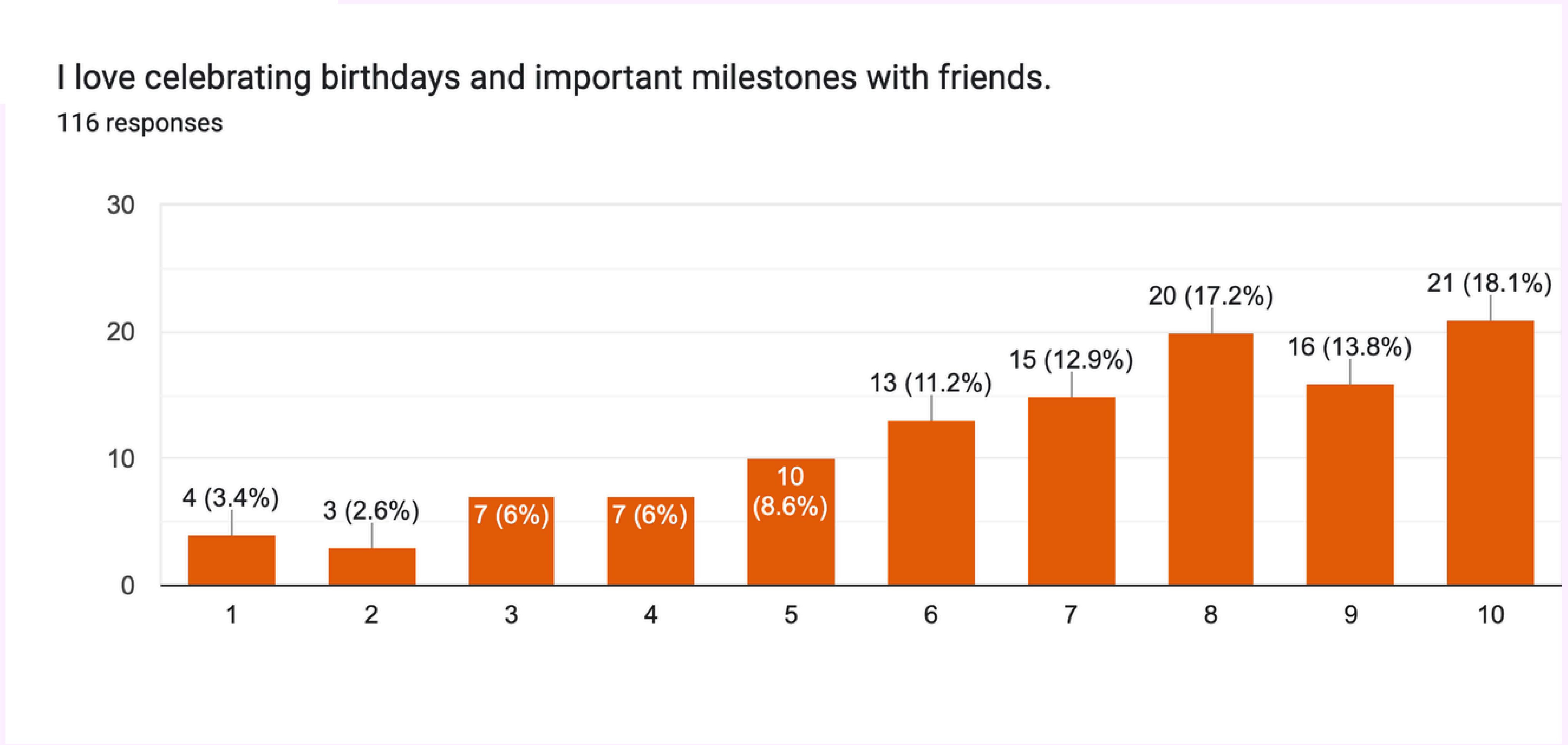


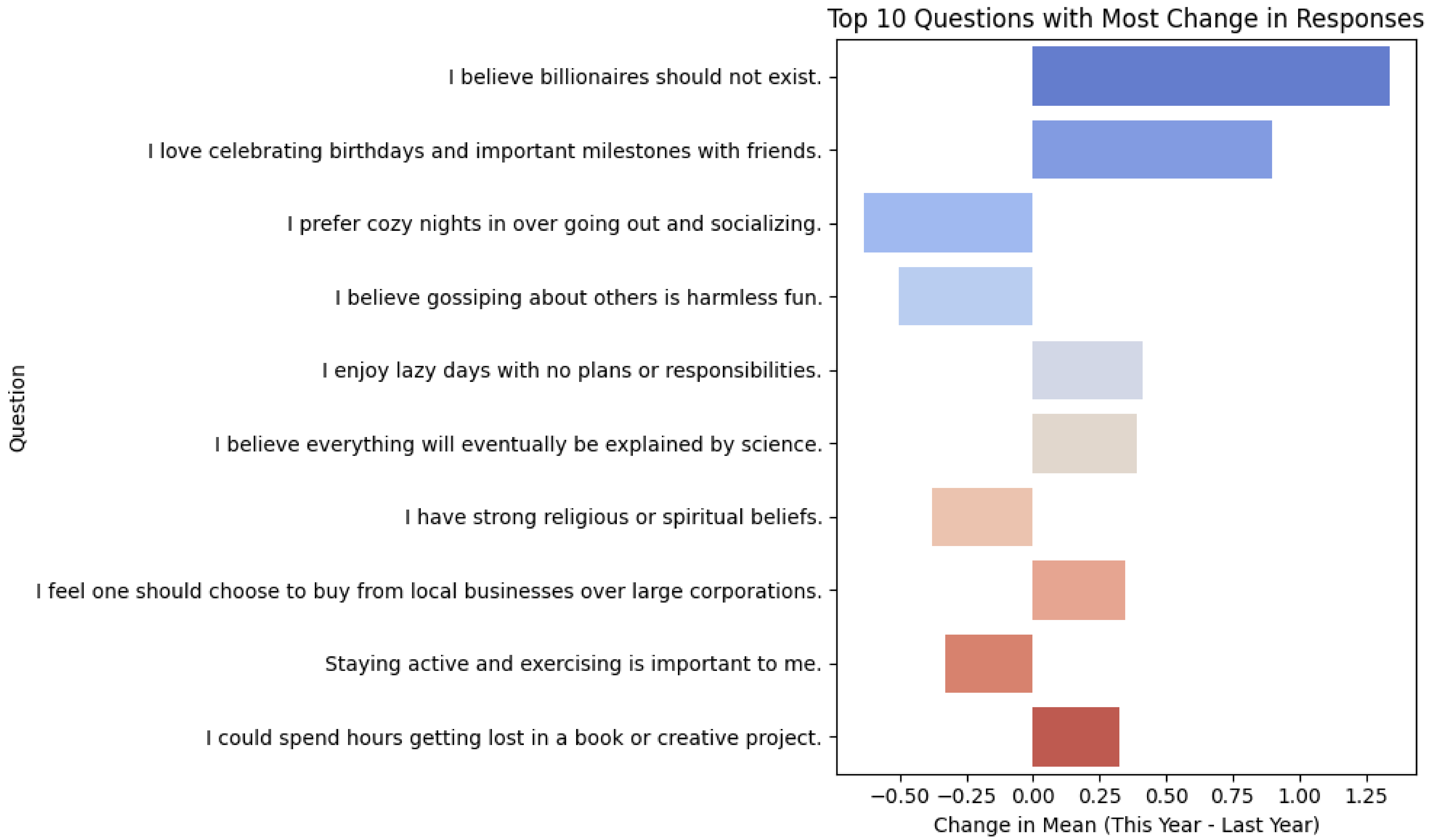
Party People



People want to party more, although by not much.

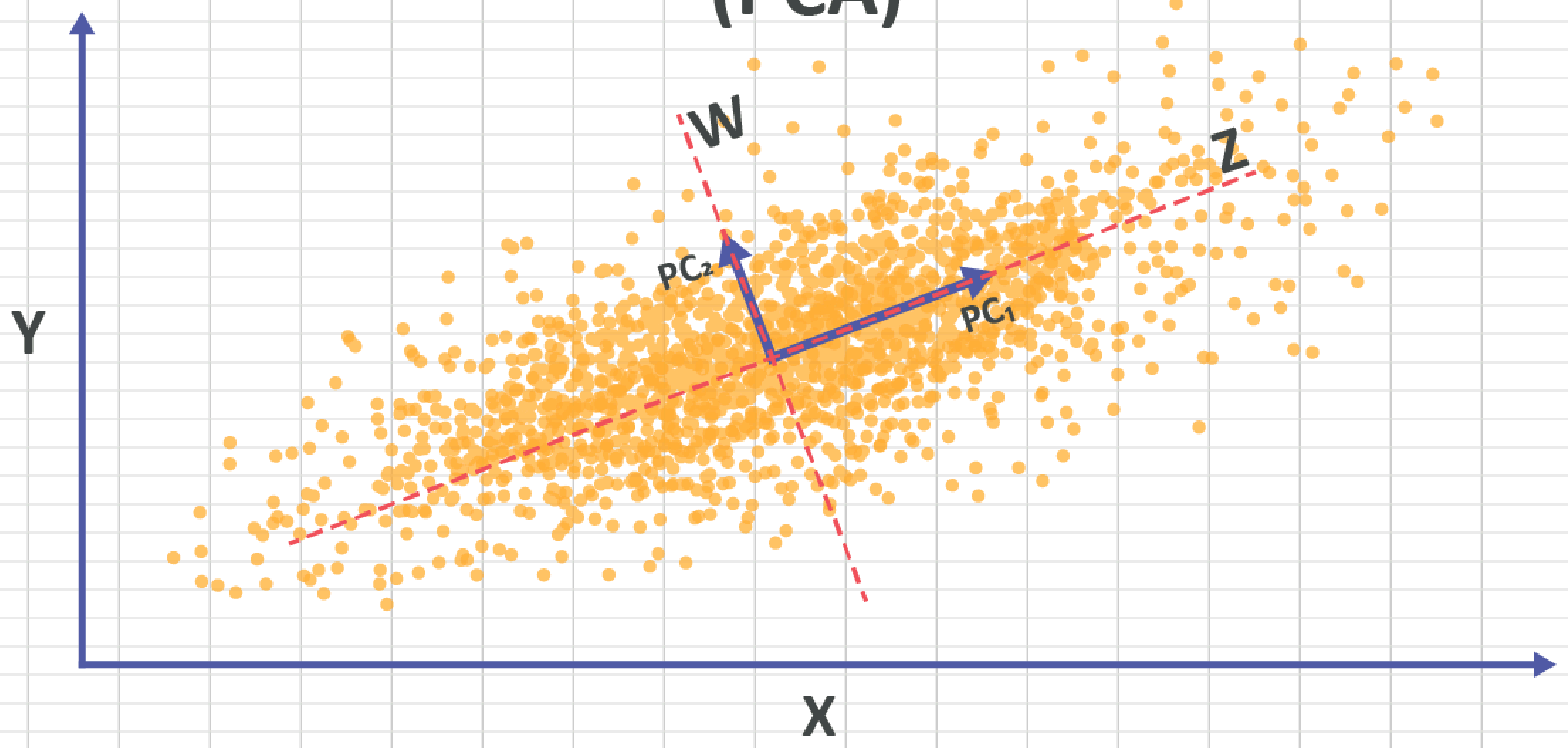
p = 0.014955232067677165
d = 0.3195806179511446

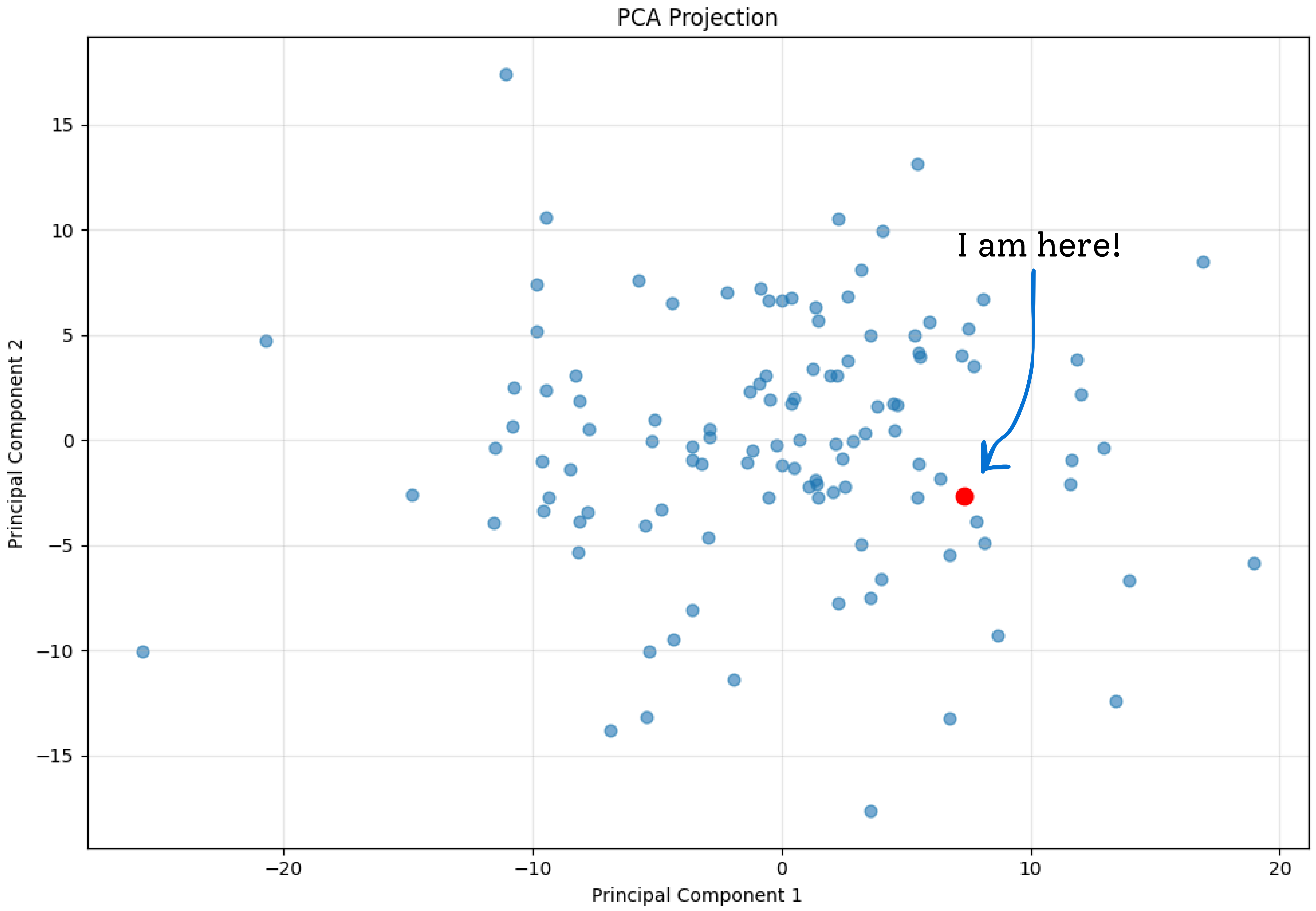


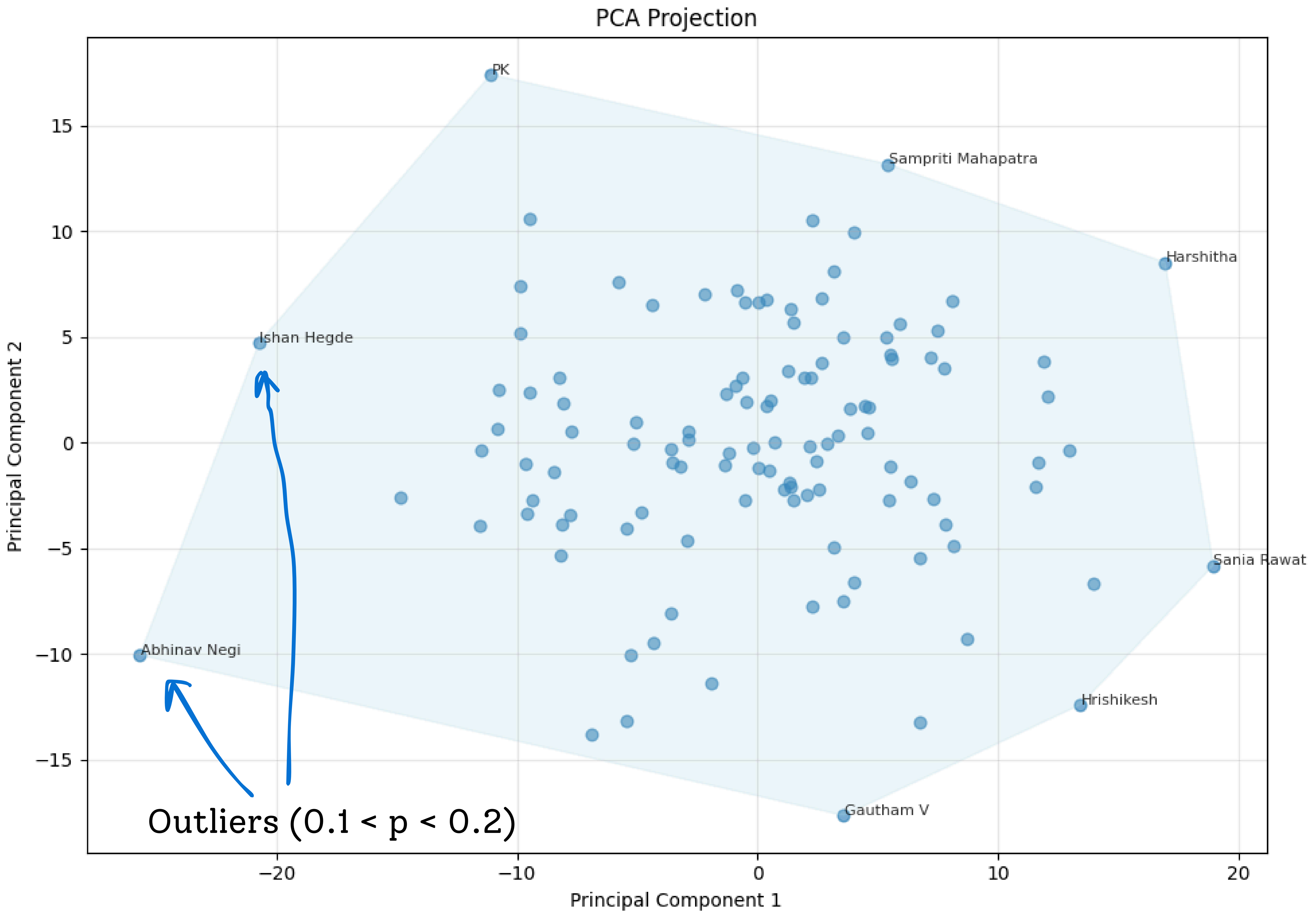

Top 10 Questions with Most Change in Responses
Question
I believe billionaires should not exist.
I love celebrating birthdays and important milestones with friends.
I prefer cozy nights in over going out and socializing.
I believe gossiping about others is harmless fun.
I enjoy lazy days with no plans or responsibilities.
I believe everything will eventually be explained by science.
I have strong religious or spiritual beliefs.
I feel one should choose to buy from local businesses over large corporations.
Staying active and exercising is important to me.
I could spend hours getting lost in a book or creative project.
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00
1.25
Change in Mean (This Year - Last Year)

Dimensionality Reduction

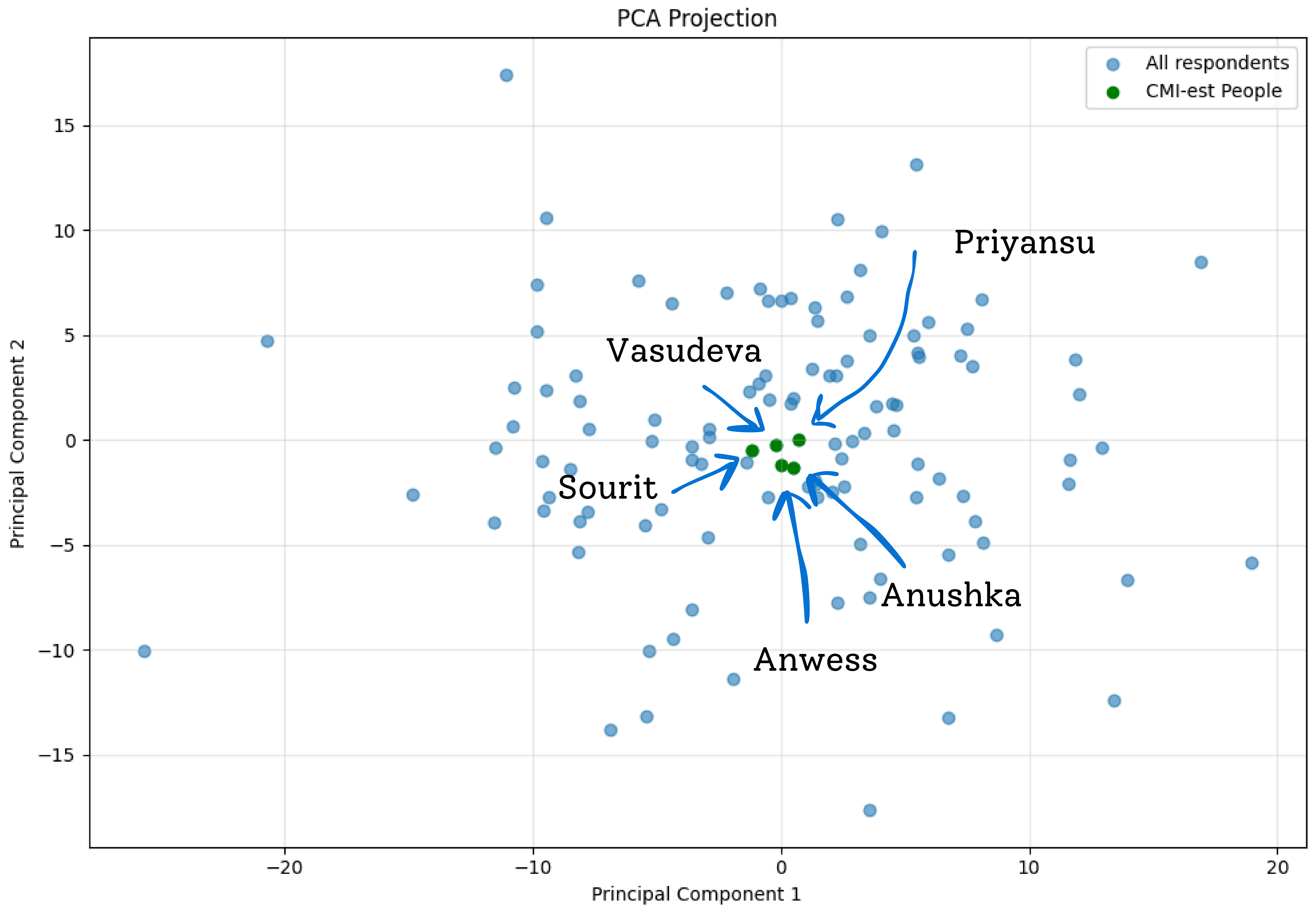
Principal Component Analysis (PCA)




PCA Projection
Principal Component 1
Principal Component 2
I am here!
−20
−10
0
10
20
15
10
5
0
−5
−10
−15



Outliers ($0.1 < p < 0.2$)


PCA Projection
All respondents
CMI-est People
Priyansu
Vasudeva
Sourit
Anushka
Anwess
Principal Component 1
Principal Component 2
−20
−10
0
10
20
15
10
5
0
−5
−10
−15

what distinguishes us?

PC1:
I love attending high-energy events like concerts, festivals, or large parties.: 0.2611
I'm comfortable with my friends drinking alcohol.: 0.2515
I like playing games that encourage sharing personal thoughts and feelings (e.g., "Never Have I Ever").: 0.2190
I love planning and hosting gatherings or parties.: 0.2114
Following a specific sport or team (e.g., cricket, football, F1) is a major hobby for me.: 0.2105
I'm comfortable with my friends using recreational drugs.: 0.2063
I enjoy following and discussing pop culture (TV shows, movies, music).: 0.2059
Expressing myself through fashion or style is important to me.: 0.2058
I'm always excited to try exotic and unusual foods.: 0.2055

PC2:
I'm comfortable with my friends using recreational drugs.: 0.2682
I'm comfortable with my friends smoking cigarettes.: 0.2563
I believe billionaires should not exist.: 0.2389
I am rarely anxious or fearful.: -0.2349
I am actively looking to make new friends and expand my social circle at this moment.: -0.2253
If matched with someone, I would be the one to send the first message to make plans: -0.2081
I prefer cozy nights in over going out and socializing.: 0.2016

06

PART 06

Conclusion

Conclusions

When I began with this, last January, I was looking for a quick pass time to get my mind of the fact that I didn’t know why I took math and CS.

To me, it always seems that most people know what they want to do and why. Number Theory, Logic, Algorithms, Algebraic Geometry etc. I was not sure.

To some extent, this project fixed that. I want to do math and CS to model, predict and understand what people do and why they do that; and use that to make better mechanisms and better choices.

Conclusions

Furthermore, It got me speaking with a lot of people in CMI and beyond.

My icebreaker with almost anyone I don’t know in CMI is “whom did you get matched with?”.

Doing this in Haskell made me understand a lot of what production Haskell looks like and why.

The data analysis work gave me a new found appreciation of the DS people (It’s not easy and I should stop calling them insects).

I have discussed this project with a lot of elder people in CMI and outside. It is almost always gotten a positive reception.

If there is to be one takeaway from this: **it is to do cool, dumb shit**.

I did not know Malay before this, and we got in touch. We both realized we shared a lot of similarities, and I think we are good friends now
-Vignesh Sangle

I was already friends with Animikha. The match brought me great joy !
-Aditi Kulkarni

It was great, found a few cultural similarities which i never thought i would find anywhere.
-Karshit Joshi

What Next?

I will run this algorithm again in 2026. Similer to Marriage Pact at Stanford, I want this to be a tradition and hence, want to maybe make it easier to use.

While friends > romance any day of the week, I really want to run the algorithm wrt romantic matching.

Similarly, I want to run the algorithm with a participant pool greater than 300. If you have friends in larger collages who might be interested, I am willing to run this for their fest.

At some point, I want to clean up the code and make it more readable. I feel it is a good tutorial for Haskell.

On the theoretical end, The average rank of the friend is still open. Similarly, the manipulation of matching with a vector-metric based preferences is not studied in much detail. Also, a lot of psychology stuff on the choice of metrics are unexplored.

Finally, I want to go to sleep (I am making this PPT way too late/early in the night/morning).

Thanks for listening to me yap!

Refrences

Gale, D., & Shapley, L. S. (1962). College Admissions and the Stability of Marriage. The American Mathematical Monthly, 69(1), 9–15. https://doi.org/10.1080/00029890.1962.11989827
Stable marriage problem. (2025, January 13). Rosetta Code. https://rosettacode.org/wiki/Stable_marriage_problem
Data.Map. (n.d.). Hackage. https://hackage.haskell.org/package/containers-0.4.0.0/docs/Data-Map.html
Error - HaskellWiki. (n.d.). https://wiki.haskell.org/Error
Type – HaskellWiki. (n.d.). https://wiki.haskell.org/index.php?title=Type
Travers, M. (2023, April 5). New Research Uncovers 24 Dimensions Of Compatibility In Long-Term Couples. Forbes. https://www.forbes.com/sites/traversmark/2023/04/05/new-research-uncovers-24-dimensions-of-compatibility-in-long-term-couples/
Marchi, A., Csajbók, Z., & Jonason, P. K. (2023). 24 ways to be compatible with your relationship partners: Sex differences, context effects, and love styles. Personality and Individual Differences, 206, 112134. https://doi.org/10.1016/j.paid.2023.112134
Marriage Pact. (n.d.). https://marriagepact.com/
Data Principles & Practices – NYU. (n.d.). Google Docs. https://docs.google.com/document/d/1Ghk1s_q65jV499_IuZIevo016ePzadXMUodB1MVlOZM
rational_lust. (2021, October 31). Marriage Pact questions list. Reddit (r/copypasta). https://www.reddit.com/r/copypasta/comments/qjlx9n/marriage_pact_questions_list/
Hutton, G. (2016). Programming in Haskell (2nd ed.). Cambridge University Press.
Irving, R. W. (1985). An efficient algorithm for the “stable roommates” problem. Journal of Algorithms, 6(4), 577–595. https://doi.org/10.1016/0196-6774(85)90033-1
Ronen, A., Hess, J. E., Belfer, Y., Mauras, S., & Eden, A. (2025). Stable Marriage: Loyalty vs. Competition. arXiv. https://doi.org/10.48550/arXiv.2501.18442
Knoblauch, V. (n.d.). Marriage Matching: A Conjecture of Donald Knuth.
Merry Algoristmas. (2014, December 5). Irving’s Algorithm for The Stable Roommate Problem [Video]. YouTube. https://www.youtube.com/watch?v=DuDvDrAXXbk
Stable roommates problem. (2025, June 17). Wikipedia. https://en.wikipedia.org/wiki/Stable_roommates_problem
Gale–Shapley algorithm. (2025, August 1). Wikipedia. https://en.wikipedia.org/wiki/Gale–Shapley_algorithm
Stable matching problem. (2025, June 24). Wikipedia. https://en.wikipedia.org/wiki/Stable_matching_problem
Brushwood, B., & Johnson, C. J. (n.d.). Pack the House!
Earth mover’s distance. (2025, September 28). Wikipedia. https://en.wikipedia.org/wiki/Earth_mover%27s_distance
Brown University. (2025, March 26). The Making of the Marriage Pact with Liam McGregor [Video]. YouTube. https://www.youtube.com/watch?v=zQnBk3jT2y4

Friendship Finder

By
Arjun Maneesh
Agarwal