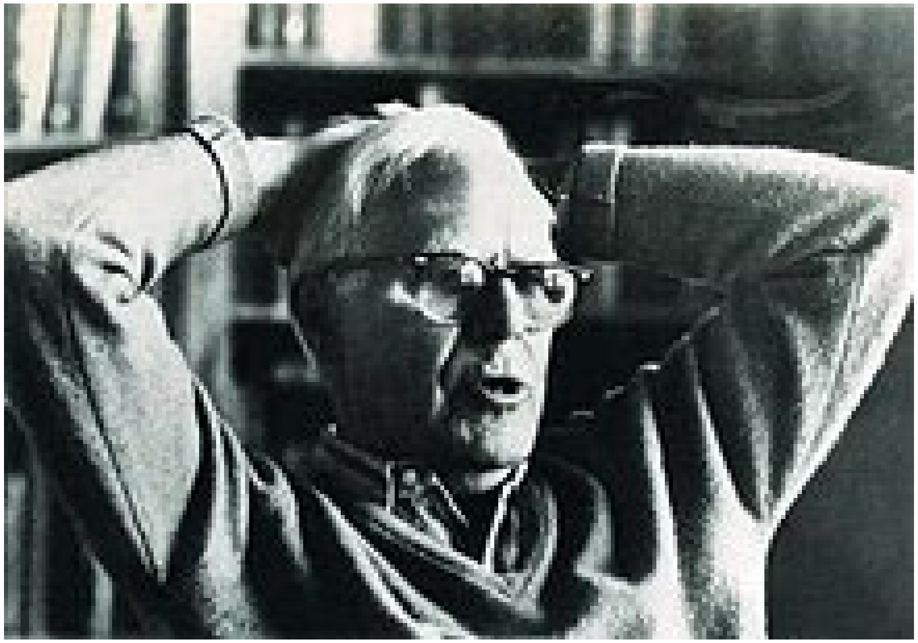
Secretary problem

- An employer has a known number – n – of applicants for a secretary position, whom are interviewed one at a time
- Employer can easily evaluate and rank applicants relative to each other but has no idea of the overall distribution of their quality
- Employer has only one chance to choose the secretary: gives yes/no answer in the end of each interview and cannot go back to rejected applicants
- How can employer maximize the probability to choose the best secretary among all applicants?



Martin Gardner (1914 – 2010)
Described the "secretary problem"
in Scientific American 1960.
was an American popular mathematics and popular science writer. Best known for "recreational mathematics": He was behind the "Mathematical Games" section in Scientific American.

Eugene Dynkin (1924 – 2014)
solved this problem in 1963.
He referred to it as a "picky bride problem"
was a Soviet and later American mathematician, member of the US National Academy of Science. He has made contributions to the fields of probability and algebra. The Dynkin diagram, the Dynkin system, and Dynkin's lemma are all named after him.

Who solved the secretary problem?

- Gardner outlined the solution in Sci Am 1960 but gave no formal proof
- Solution by Lindey was published in 1961:
  Lindey, D. V. (1961). Dynamic programming and decision theory. Appl. Statist. 10 39-51
- Dynkin's paper was published in 1963:
  Dynkin, E. B. (1963). The optimum choice of the instant for stopping a Markov process. Soviet Math. Dokl. 4 627-629
- When the celebrated German astronomer, Johannes Kepler (1571-1630), lost his first wife to cholera in 1611, he set about finding a new wife
- He spent 2 years on the process, had 11 candidates and married the 5th candidate (11/e~4 so he married the first after)

Thomas S Ferguson, Statistical Science 1989, Who Solved the Secretary Problem?

What should the employer do?

- Employer does not know the distribution of the quality of applicants and has to learn it on the fly
- Algorithm: look at the first ***r-1*** applicants, remember the best among them
- Hire the first among next ***n-r+1*** applicants who is better than the best among the first ***r*** applicants
- How to choose ***r***?
- When ***r*** is too small – not enough information: the best among *r* is not very good. You are likely to hire a bad secretary
- When ***r*** is too large (e.g. ***r=n-1***) – you procrastinated for too long! You have almost all the information, but you will have to hire the last applicant who is (likely) not particularly good

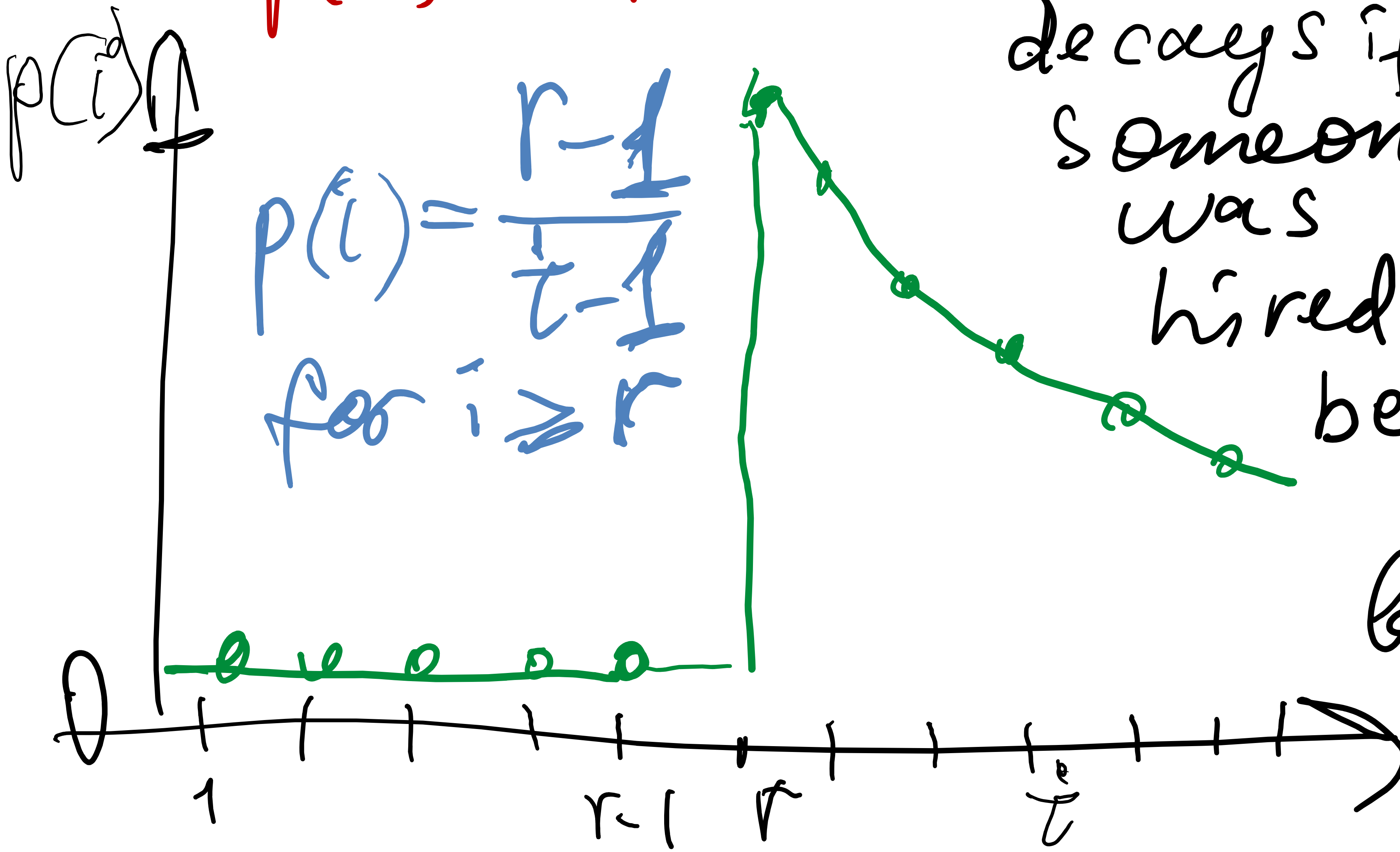
$$
\begin{aligned}
P(r) &= \sum_{i=1}^{n} P\left(\text{applicant } i \text{ is selected} \cap \text{applicant } i \text{ is the best}\right) \\
&= \sum_{i=1}^{n} P\left(\text{applicant } i \text{ is selected} \middle| \text{applicant } i \text{ is the best}\right) \times P\left(\text{applicant } i \text{ is the best}\right)
\end{aligned}
$$

$1/n$

How to calculate?

Probability of hiring the best candidate if they are #i in the queue

$p(i) = 0$ for $i < r$



$p(i) = \frac{r-1}{t-1}$ for $i \geq r$

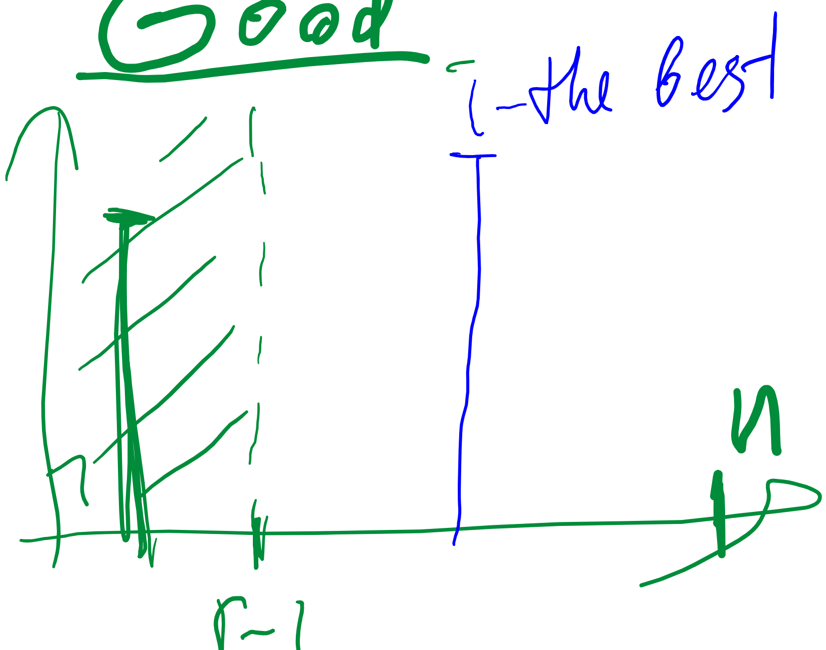
decays if someone was hired before the best

Look at $i-1$ candidates before the best

$$\text{Prob} = \frac{r-1}{i-1}$$

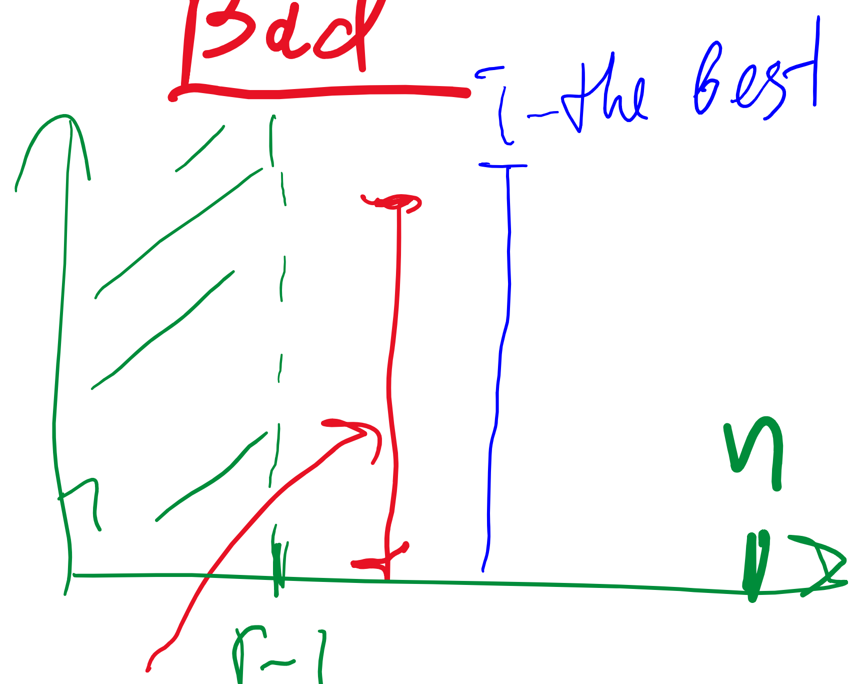
$$\text{Prob} = \frac{i-r}{i-1}$$

Good



the best among $i-1$

Bad



the best among $i-1$

$$
\begin{aligned}
P(r) &= \sum_{i=1}^{n} P\left(\text{applicant } i \text{ is selected} \cap \text{applicant } i \text{ is the best}\right) \\
&= \sum_{i=1}^{n} P\left(\text{applicant } i \text{ is selected} \middle| \text{applicant } i \text{ is the best}\right) \times P\left(\text{applicant } i \text{ is the best}\right) \\
&= \left[\sum_{i=1}^{r-1} 0 + \sum_{i=r}^{n} P\left(\begin{array}{l}\text{the best of the first } i-1 \text{ applicants} \\ \text{is in the first } r-1 \text{ applicants}\end{array} \middle| \text{applicant } i \text{ is the best}\right)\right] \times \frac{1}{n} \\
&= \sum_{i=r}^{n} \frac{r-1}{i-1} \times \frac{1}{n} \quad = \quad \frac{r-1}{n} \sum_{i=r}^{n} \frac{1}{i-1}.
\end{aligned}
$$

$$P(r) = \frac{r-1}{n}\sum_{i=r}^{n}\frac{1}{i-1}.$$

Letting n tend to infinity, writing x as the limit of r/n, using t for i/n and dt for $1/n$,

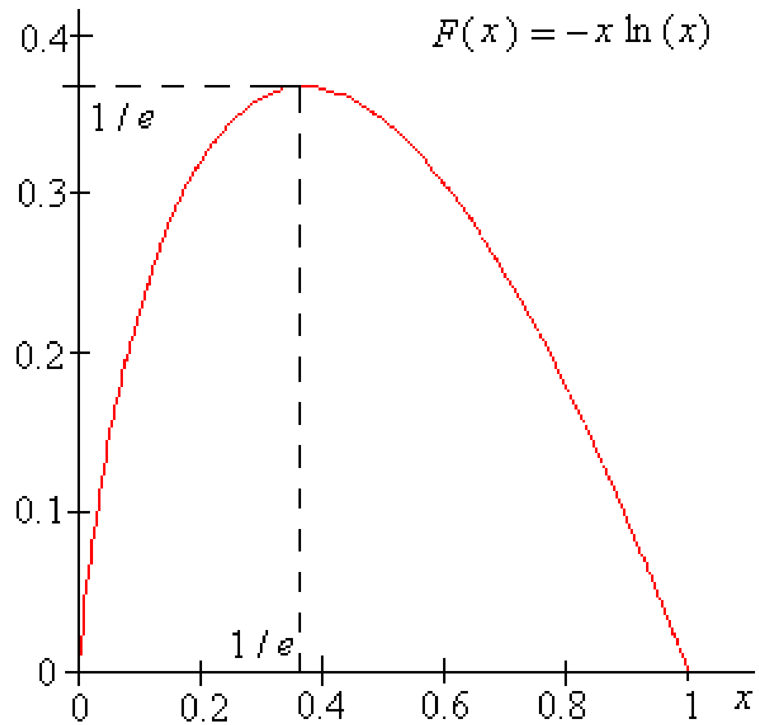
$$P(x) = x\int_x^1 \frac{1}{t}\,dt = -x\ln(x)\ .$$

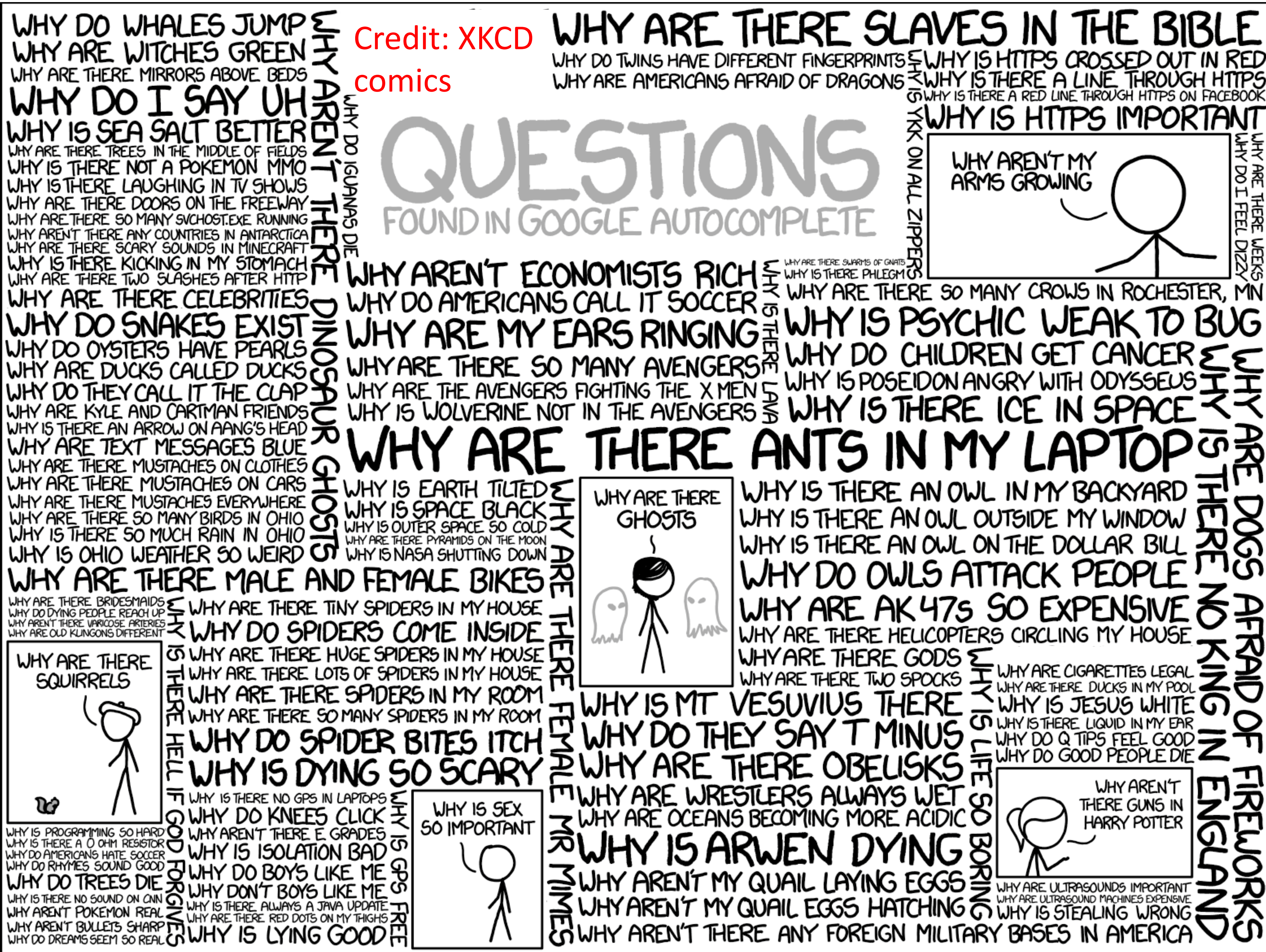
dP(x)/dx=-ln(x)-1

-ln(x)-1=0*

x=1/e=0.3679*

Probability of picking the best applicant is also *1/e=0.3679*





Credit: XKCD comics

Simpson’s paradox

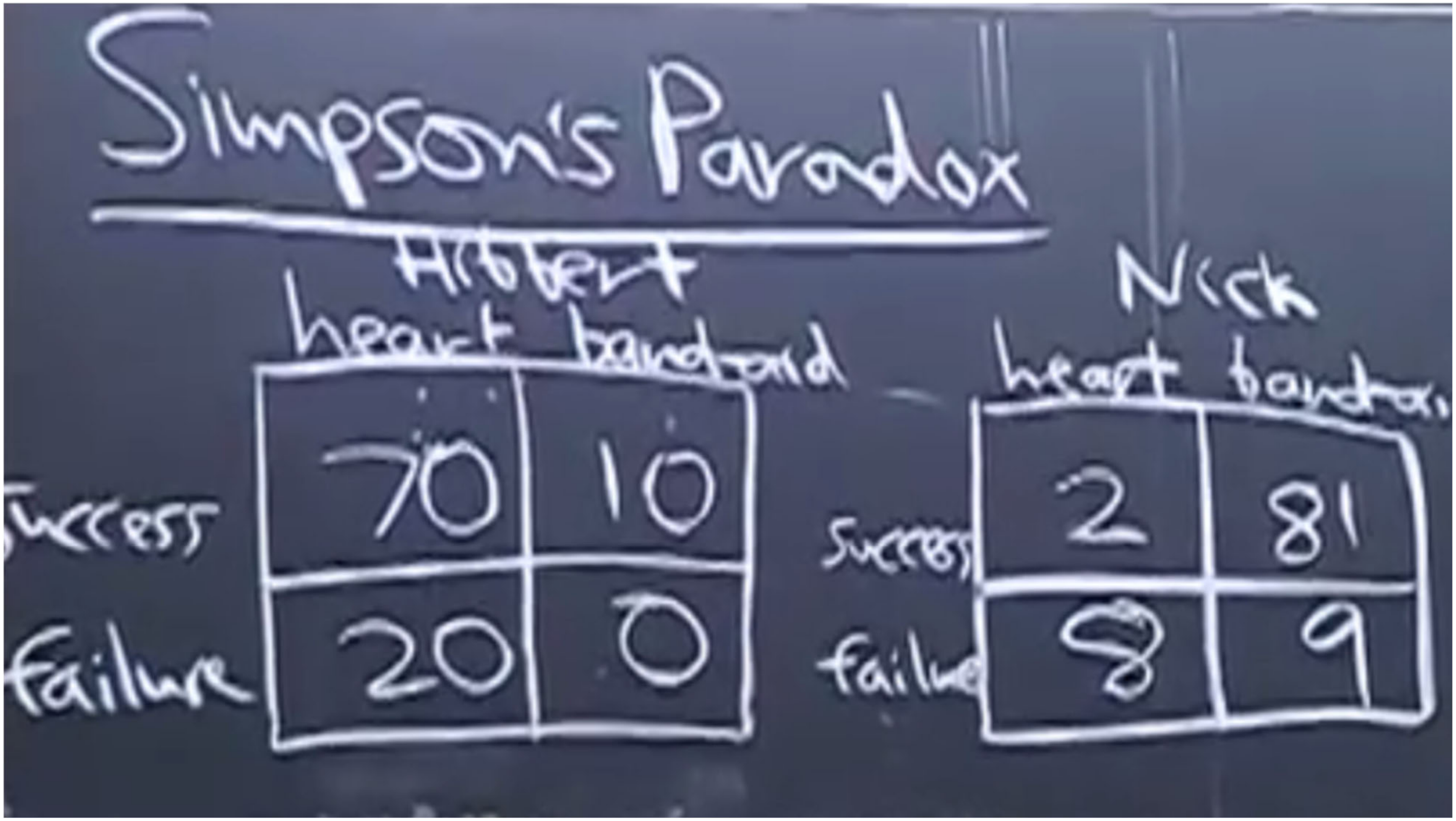
Edward Hugh Simpson
(10 December 1922 – 5 February 2019)
was a British codebreaker, statistician
and civil servant.

"The Interpretation of Interaction
in Contingency Tables“, Journal of the Royal Statistical Society, 1951

Is it possible for one doctor to have a **higher** success rate than another doctor in **every** type of treatment he performs but to have a **lower** **overall** success rate across all treatment types?

Dr. Hibbert

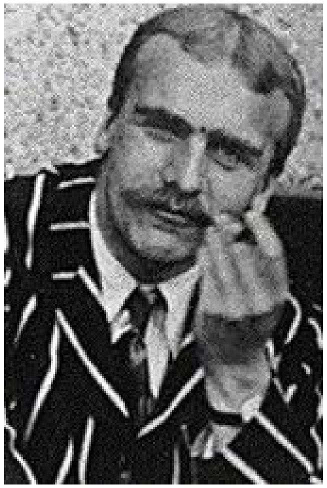
Dr. Nick



Dr. Hibbert: success rate =80%
Dr. Nick: success rate =83%

Simpson's paradox might explain altruism

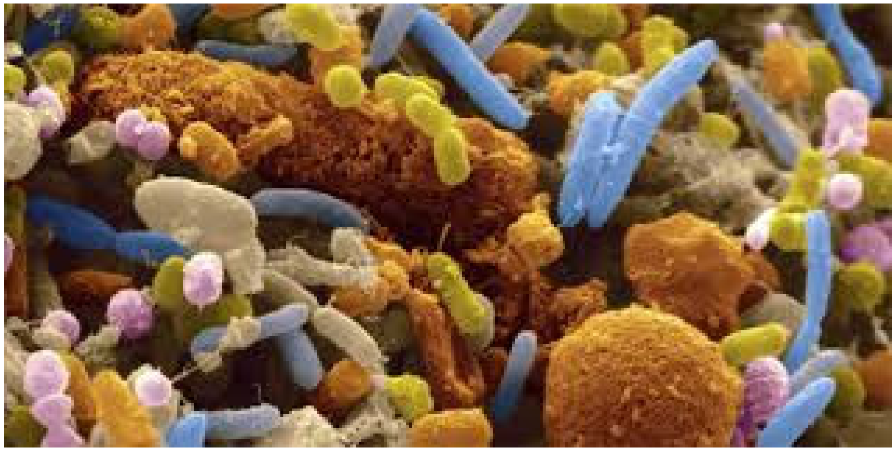
- Darwinian evolution has a problem with altruism
- "Selfish genes" do not care about others
- J. B. S. Haldane, (1892-1964) British geneticist, evolutionary biologist



- When asked if he would give his life to save a drowning brother answered: "No, but I would to save two brothers or eight cousins"
- Altruism in some insect colonies like ants is because they are all genetically similar.

Altruism in bacteria

- Bacteria live in communities in close proximity to each other
- Individual bugs spend significant resources to produce extracellular molecules, excrete them outside of the cell to share with others. **That slows their growth**
  - Examples: extracellular enzymes, biofilm components, antimicrobial and anti-immune agents
- Cheaters have faster growth rate
  - They can take over by not producing any shared molecules
- **Evolutionary paradox: how bacteria can be altruistic?**



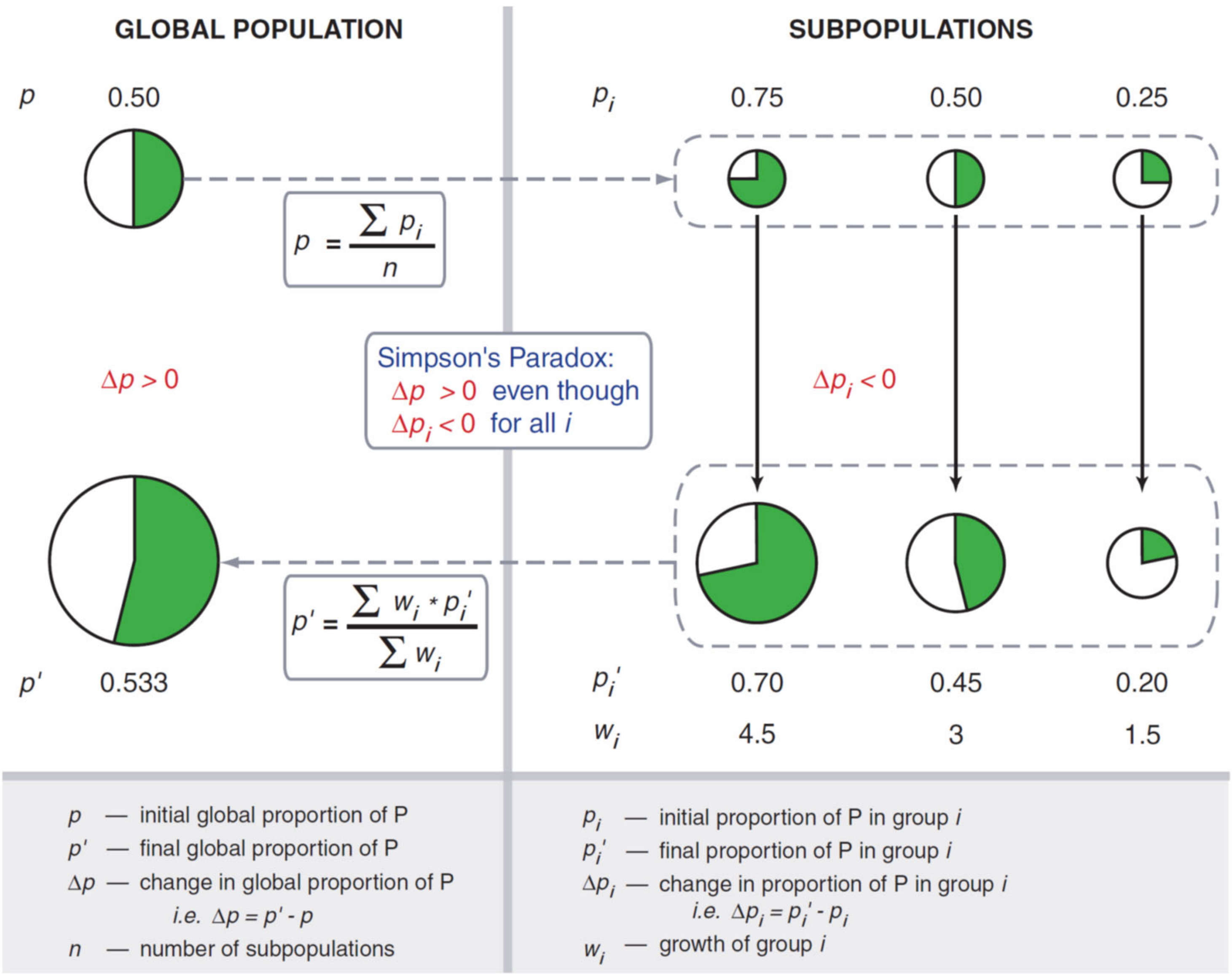
Chuang, Rivoire, and Leibler's answer

Simpson's Paradox in a Synthetic Microbial System

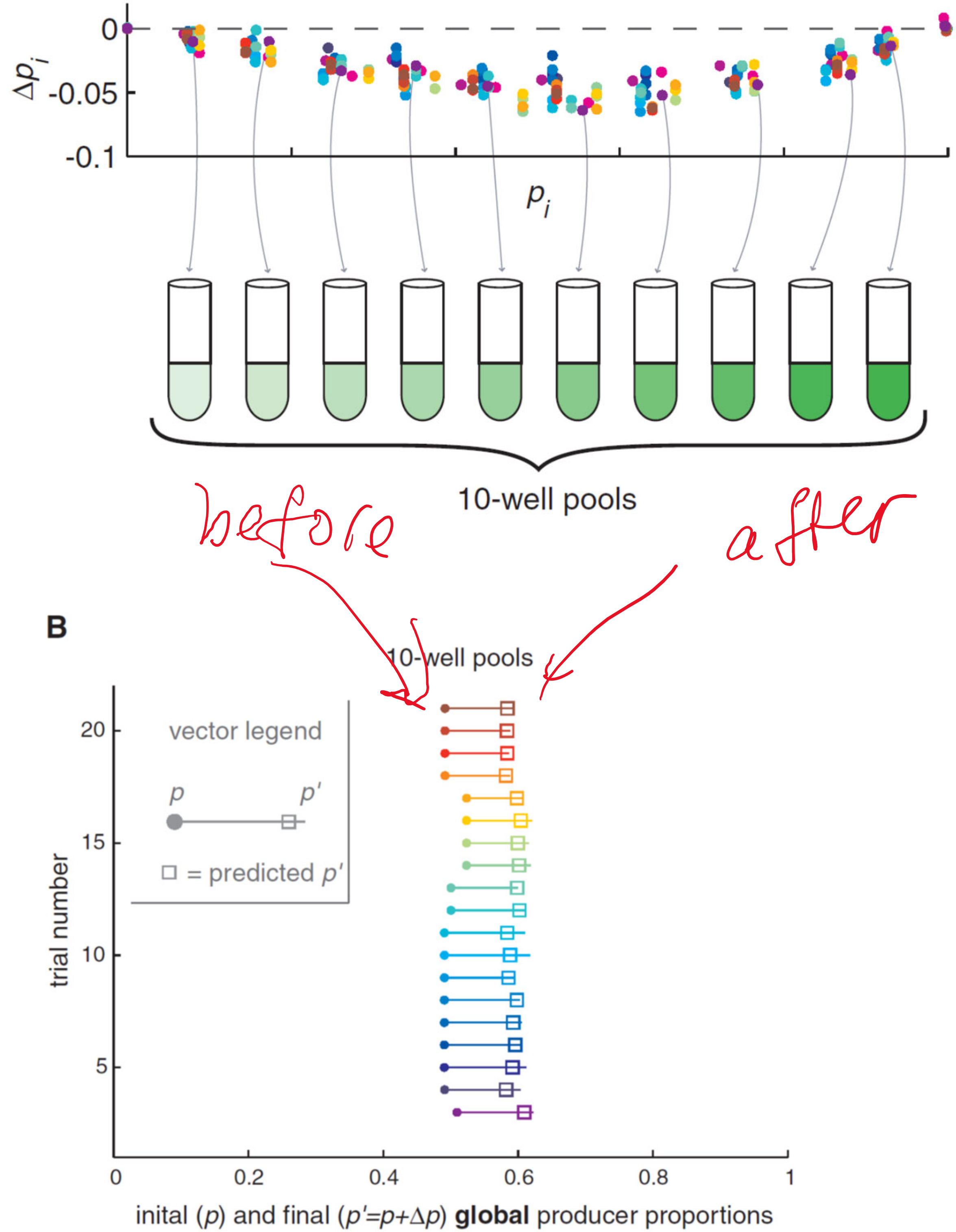
John S. Chuang,* Olivier Rivoire, Stanislas Leibler

The maintenance of "public" or "common good" producers is a major question in the evolution of cooperation. Because nonproducers benefit from the shared resource without bearing its cost of production, they may proliferate faster than producers. We established a synthetic microbial system consisting of two *Escherichia coli* strains of common-good producers and nonproducers. Depending on the population structure, which was varied by forming groups with different initial compositions, an apparently paradoxical situation could be attained in which nonproducers grew faster within each group, yet producers increased overall. We show that a simple way to generate the variance required for this effect is through stochastic fluctuations via population bottlenecks. The synthetic approach described here thus provides a way to study generic mechanisms of natural selection.

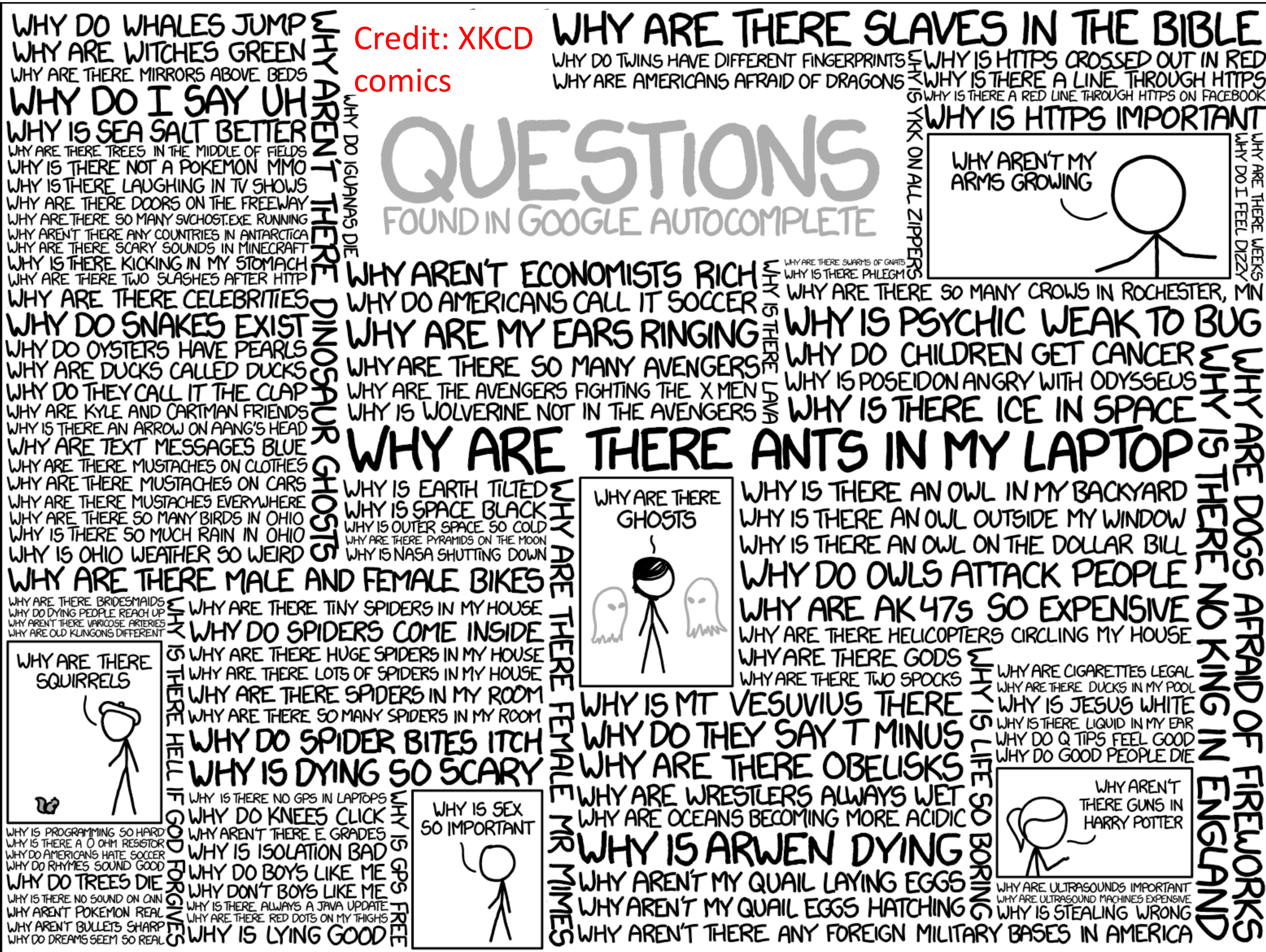
- The common good was a membrane-permeable Rhl autoinducer molecule rewired to activate antibiotic (chloramphenicol; Cm) resistance gene expression.



Fraction of altruists in each of individual test tubes dropped

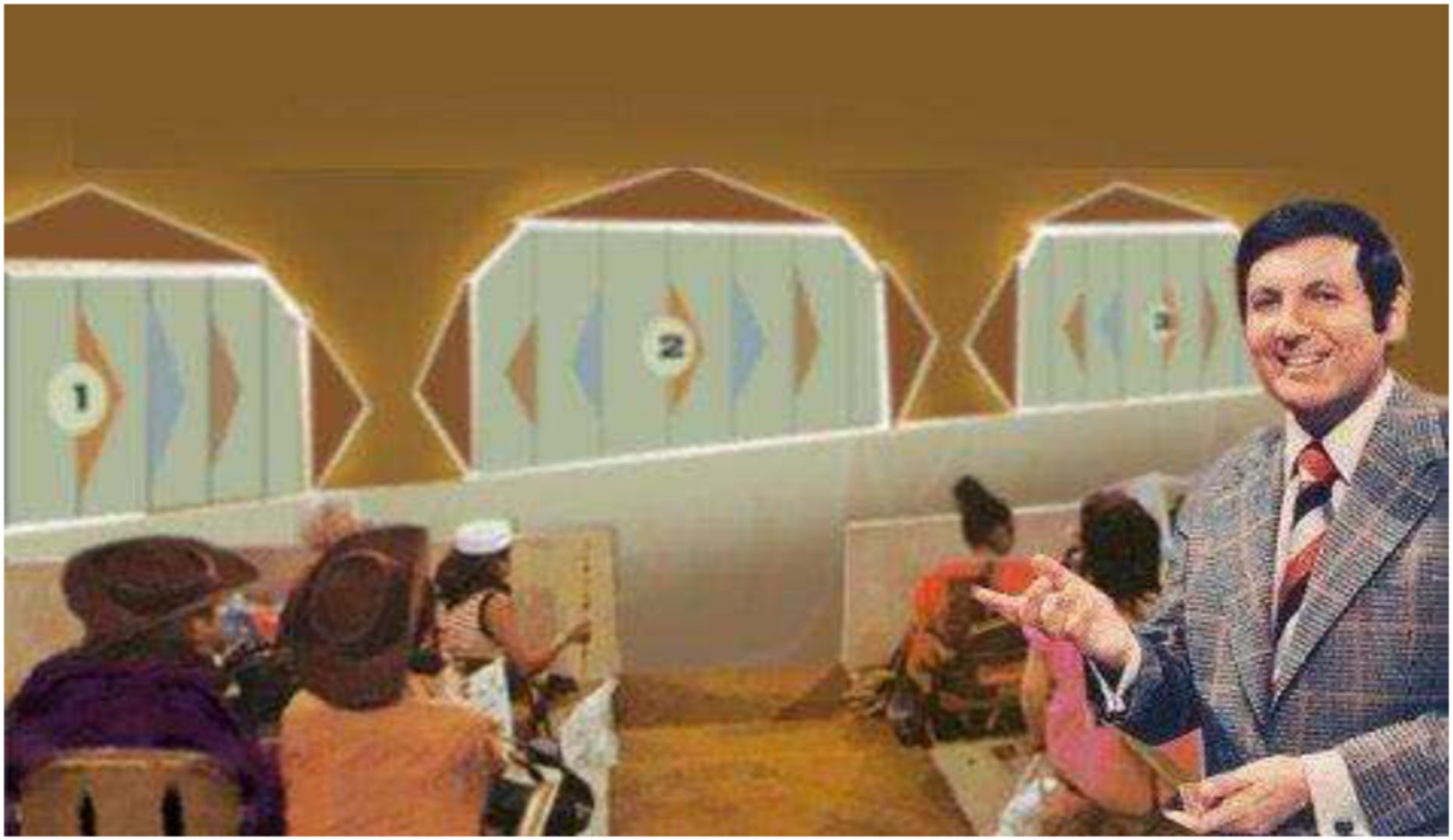


Yet the overall fraction of altruists in all test tubes combined increased



Credit: XKCD comics

Monty Hall problem



Monty Hall OC, OM (born Monte Halparin)
August 25, 1921 – September 30, 2017
was a Canadian-American game show host, producer, and philanthropist

Game show “Let’s Make a Deal” aired 1963-now

Monty Hall problem

- In *Make a Deal* there are three closed doors. Behind a random one of these doors is a car; behind the other two are goats. The contestant does not know where the car is, but Monty Hall does.
- After the contestant picks a door Monty always opens one of the remaining doors, one he knows does not hide the car. If the contestant has already chosen the door with the car behind, Monty is equally likely to open either of the two remaining doors.
- After Monty has shown a goat behind the door that he opens, the contestant is always given the option to switch doors.
- What is the probability of winning the car under the switching and non-switching strategies?

Monty Hall problem. What strategy gives you a better chance to win the car?

A. Better to switch doors
B. Better not to switch doors
C. Switching does not matter
D. The answer depends on the phase of the moon
E. I don't know

Get your i-clickers

Monty Hall problem. What strategy gives you a better chance to win the car?

A. Better to switch doors
B. Better not to switch doors
C. Switching does not matter
D. The answer depends on the phase of the moon
E. I don't know

Get your i-clickers

Don’t feel bad if you guessed wrong

- When first presented with the Monty Hall problem an overwhelming majority of people assume that each door has an equal probability and conclude that switching does not matter
- Out of 228 subjects in one study, only 13% chose to switch
- Paul Erdős, one of the most prolific mathematicians in history, remained unconvinced until he was shown a computer simulation confirming the predicted result
- Pigeons repeatedly exposed to the problem show that they rapidly learn always to switch, unlike humans

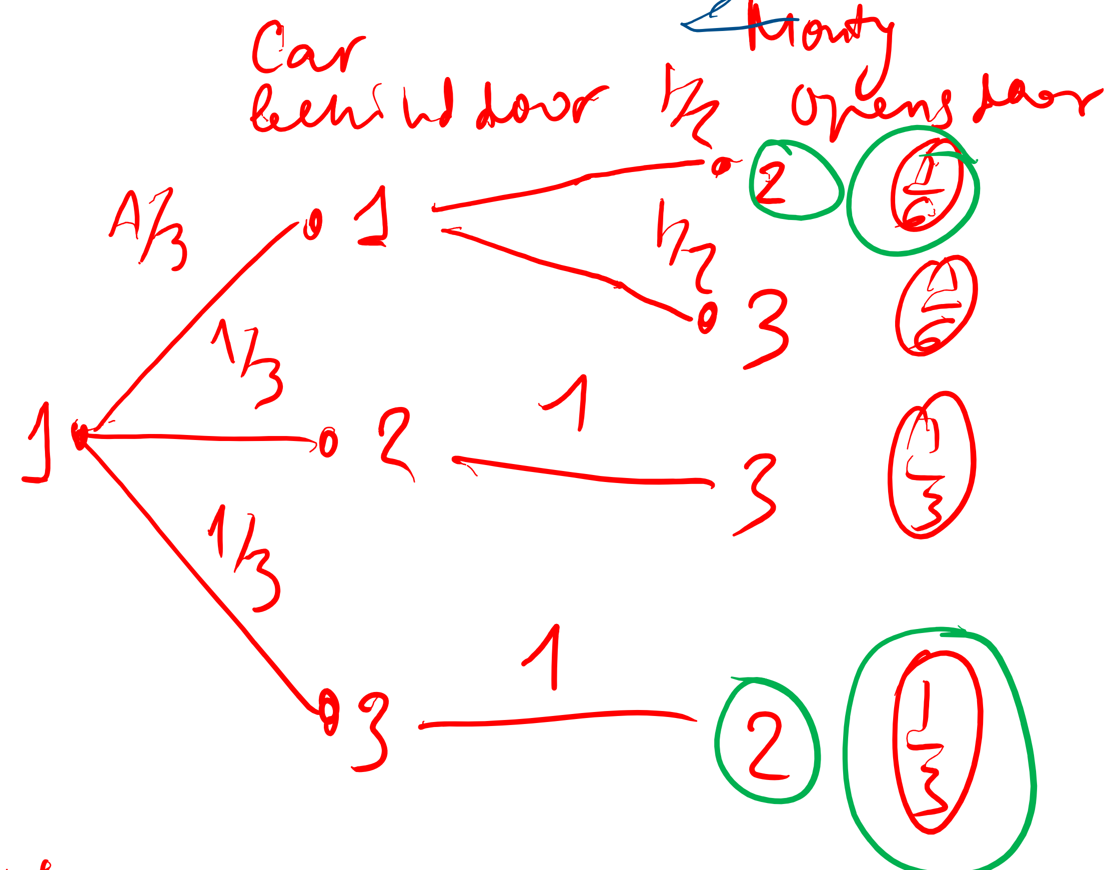
Solution #1 (intuitive)

- With Prob=1/3 you guess the car door right: you loose the car if you switch
- With Prob=2/3 you got it wrong and picked a goat door. Then Monty opens another goat door. What is left is the car door. You win the car if you switch!

Solution #2.
Tree & conditional probabilities

Solution #2.
Tree & conditional probabilities

Contestant chooses door 1



conditional probability

$$P(\text{Monty opens door 2} \mid \text{car behind door 1}) = \frac{1}{2}$$

If monty opened door 2

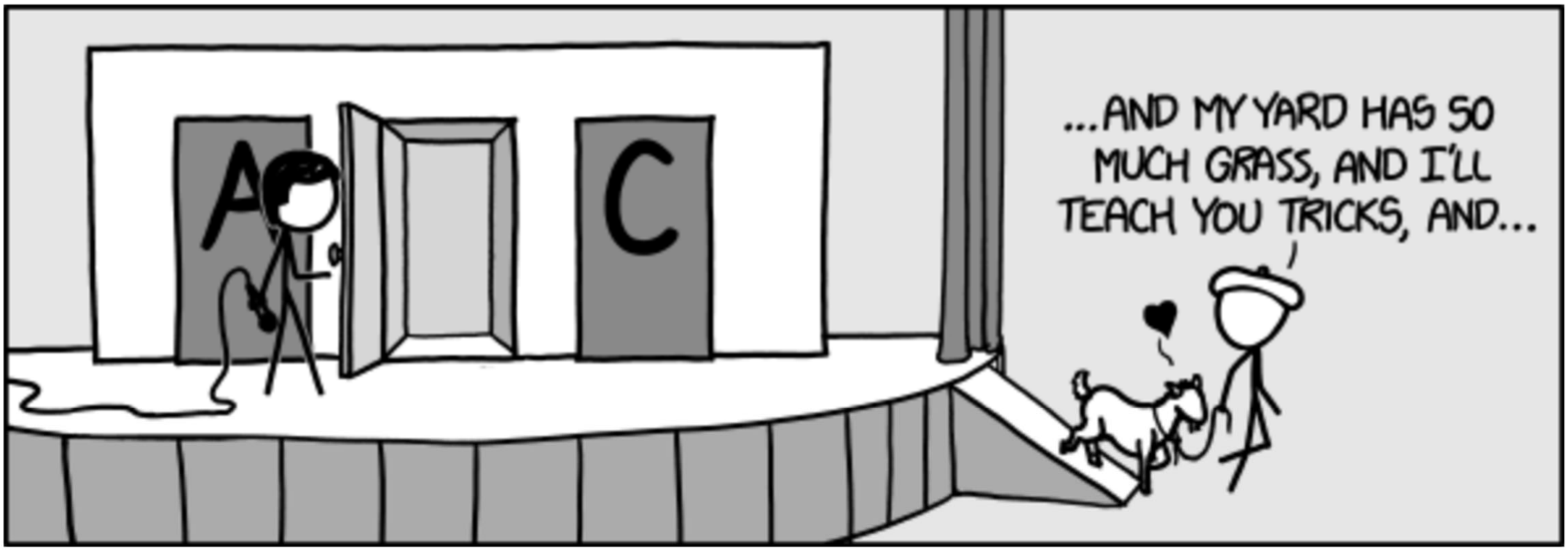
$$\text{Prob}(\text{win by switching}) = \frac{\frac{1}{3}}{\frac{1}{3}+\frac{1}{6}} = \frac{2}{3}$$

$$\text{Prob}(\text{win by staying}) = \frac{1/6}{1/3+1/6} = \frac{1}{3}$$

A more detailed tree diagram

- Shinyapp website https://dacalderon.shinyapps.io/montyhall/

Thanks to my former BIOE 505 student,
Alejandra Zeballos Castro, for bringing it to my attention



comic credits: xkcd

Let’s check the theory by playing the dame

Go to https://dacalderon.shinyapps.io/montyhall/

- Right side will play “switch the door” strategy
- Left side of the auditorium will play “do not switch the door” strategy
- Each person should play at least 30 rounds (more is better)
- Multiple players at the same table add up your numbers of games played and separately the number of wins
- In the end I will **add up the numbers from all tables**

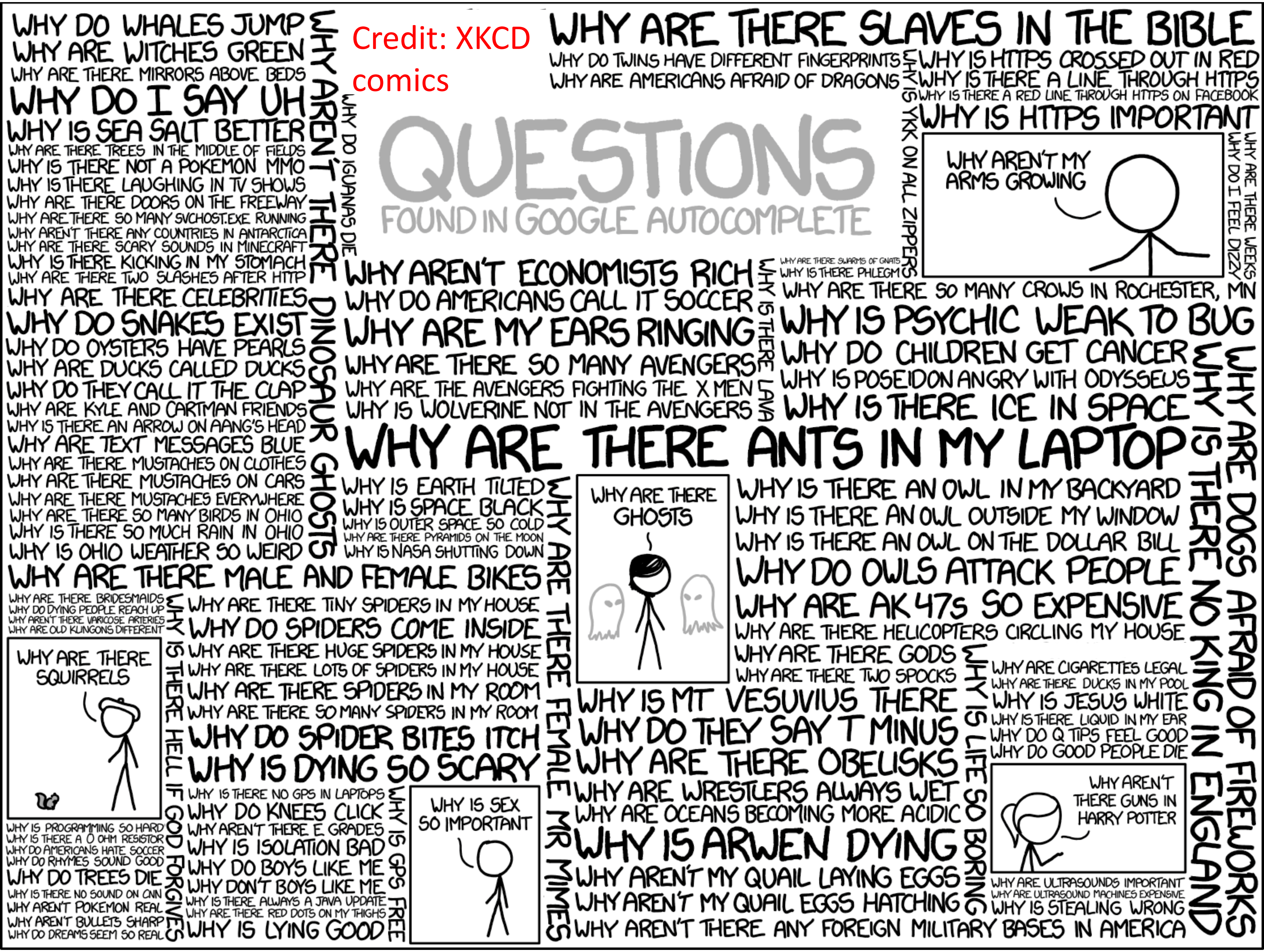
	Switch strategy			No switch strategy	
Table	Played	Won	Table	Played	Won
A	30	20	D	96	33
B	25	11	E	80	25
C	80	62	F	80	16
	135	93		256	74
P(win)	0.688889			0.289063	

Let’s check with more random experiments

- **Stats=??;**
- **%set Stats large...**
- **switch_count=0; noswitch_count=0; %set 0 at the beginning**
- **for n = 1:Stats**
- **a = randperm(3); %Monty places two goats and the car at random**
- **%a(1) -goat, a(2) -goat, a(3) - car**
- **i= floor(3.*rand)+1; %you select the door!**
- **% SWITCH STRATEGY**
- **if(i == a(1)) switch_count=switch_count+??; %a(2)-opened, switch to a(3), car!**
- **elseif (i == a(2)) switch_count = switch_count + ??;%a(1) opened, switch to a(3), car!**
- **else switch_count = switch_count + ??; %a(1)/a(2) opened, switch to a(2)/a(1), no car :-(**
- **end**
- **% NO SWITCH STRATEGY**
- **if(i == a(1)) noswitch_count = noswitch_count + ??; %a(2)-opened, no car :-(**
- **elseif (i==a(2)) noswitch_count = noswitch_count + ?? %a(1)-opened, no car :-(**
- **else noswitch_count = noswitch_count + ??; %a(1) or a(2)-opened, car!**
- **endend;**
- **disp('probability to win a car if switched doors=');**
- **disp(num2str(switch_count./??)); %# of cars with switching**
- **disp('probability to win a car if did not switch doors=');**
- **disp(num2str(noswitch_count./??)); %# of cars w/o switching**

Matlab program

- B=10000; %set B large...
- cars=0; carn=0; %set 0 at the beginning
- for i = 1:B
- a = randperm(3); %Monty places two goats and the car at random
- %a(1) -goat, a(2) -goat, a(3) - car
- i= floor(3.*rand)+1; %you select the door!
- % SWITCH STRATEGY
- if(i == a(1)) cars=cars+1; %a(2)-opened, switch to a(3), car!
- elseif (i == a(2)) cars = cars + 1 ;%a(1) opened, switch to a(3), car!
- else cars = cars + 0; %a(1)/a(2) opened, switch to a(2)/a(1), no car!
- end
- % SWITCH STRATEGY
- if(i == a(1)) carn = carn + 0; %a(2)-opened, no car
- elseif (i==a(2)) carn = carn + 0; %a(1)-opened, no car
- else carn = carn + 1; %a(1) or a(2)-opened, car!
- end
- end;
- disp('probability to win a car if switched doors=');
- disp(num2str(cars./B)); %# of cars with switching
- disp('probability to win a car if did not switch doors=');
- disp(num2str(carn./B)); %# of cars w/o switching



Credit: XKCD comics