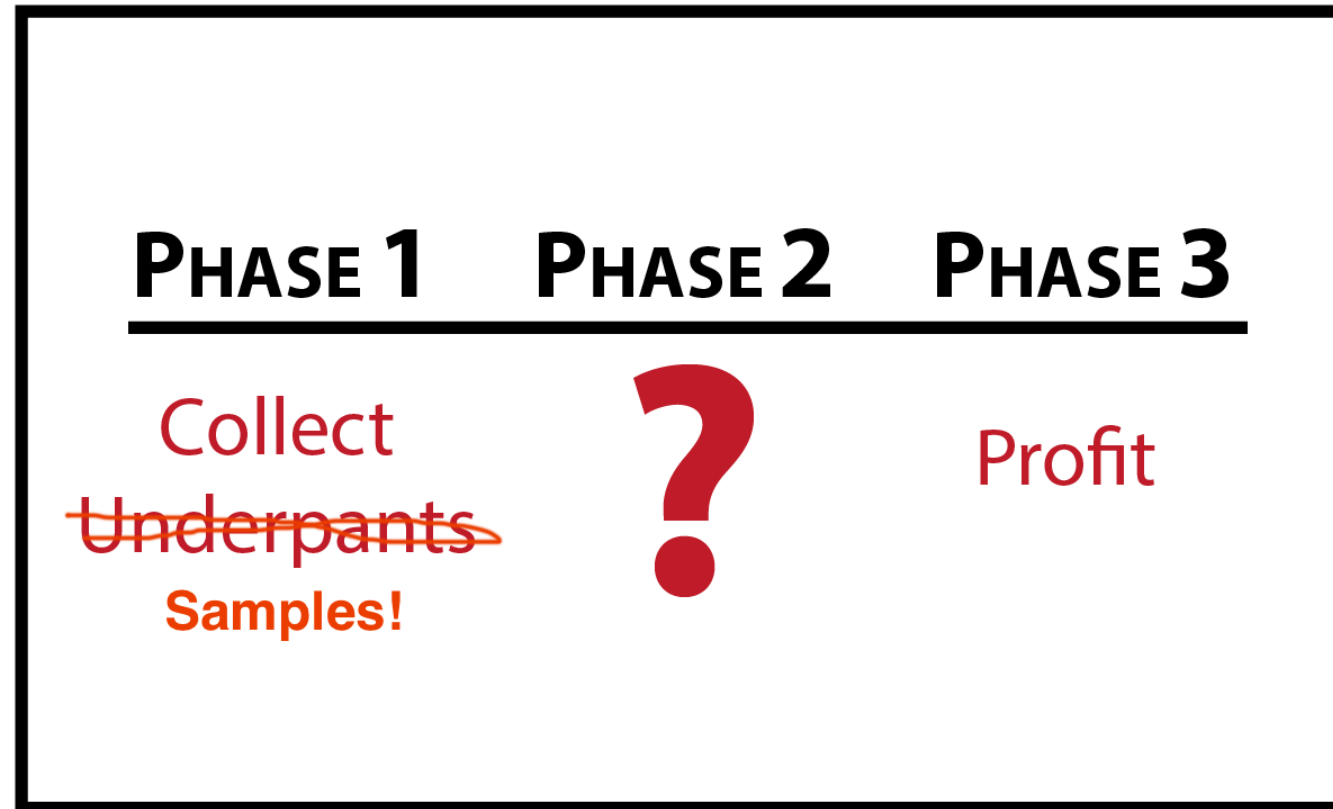


Joy of Simulation



For Fun ... and Profit!

Who is person?

Vincent D. Warmerdam

Data Person @ GoDataDriven in Amsterdam

What we do?

Please bring out your laptops.

We are going to do an experiment in the beginning.

Today

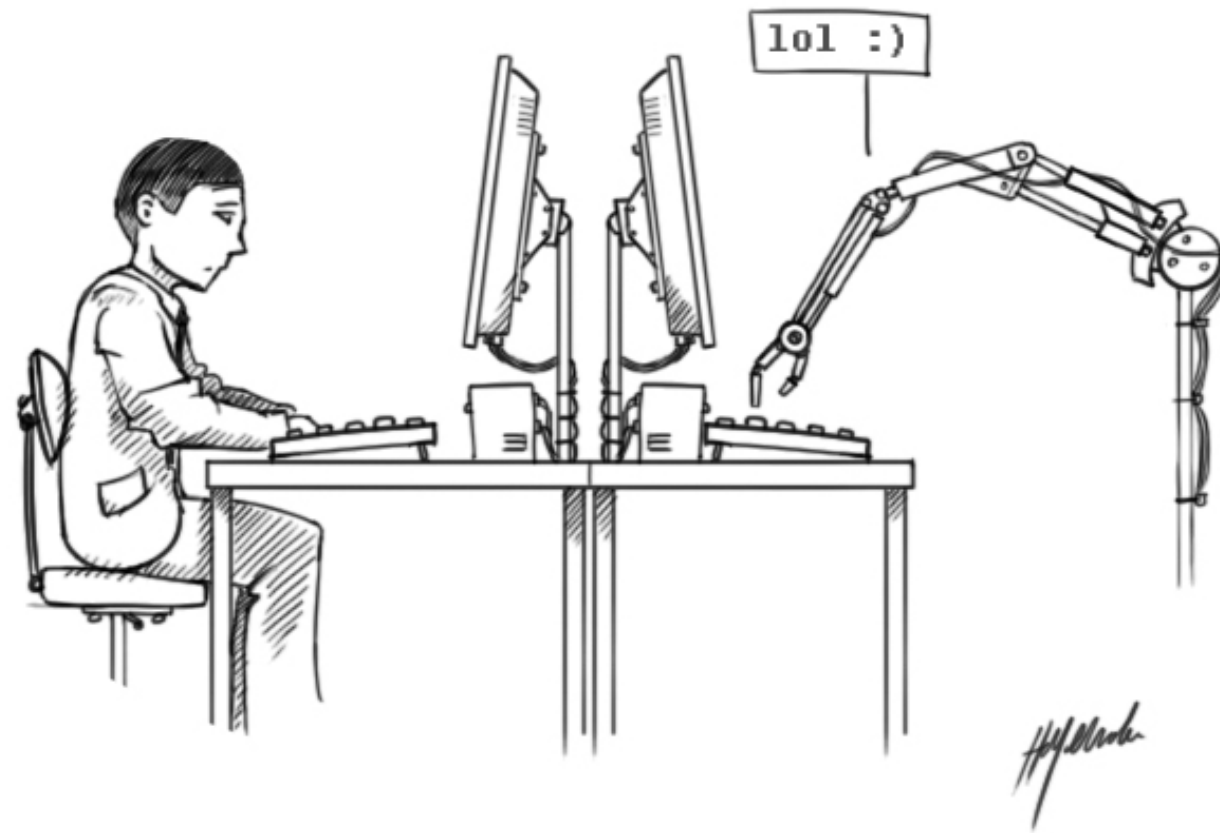
- randomness: what it is, what it isn't (5 min)
- explain sampling for modelling inference (1 min)
- better tactics for monopoly (5 min)
- sell lego minifigures/ebay (5 min)
- sampling as an optimisation tactic (5 min)
- outsourcing creativity (2 min)
- pokemon related subject (8 min)

Randomness

Before talking about what it is.

We should make sure what it is not.

What is not randomness?



Are you ready for an experiment?

What is not randomness?

Go to **<http://koaning.io/>**.

Click the blogpost named **human entropy**.

Await (or read) instructions.

Human Entropy ...

... is terrible!

This is why we prefer to use a computer to help us think about probability. We could also use maths but often using a computer is just easier.

The goal of this talk is to convince you of this via some fun examples.

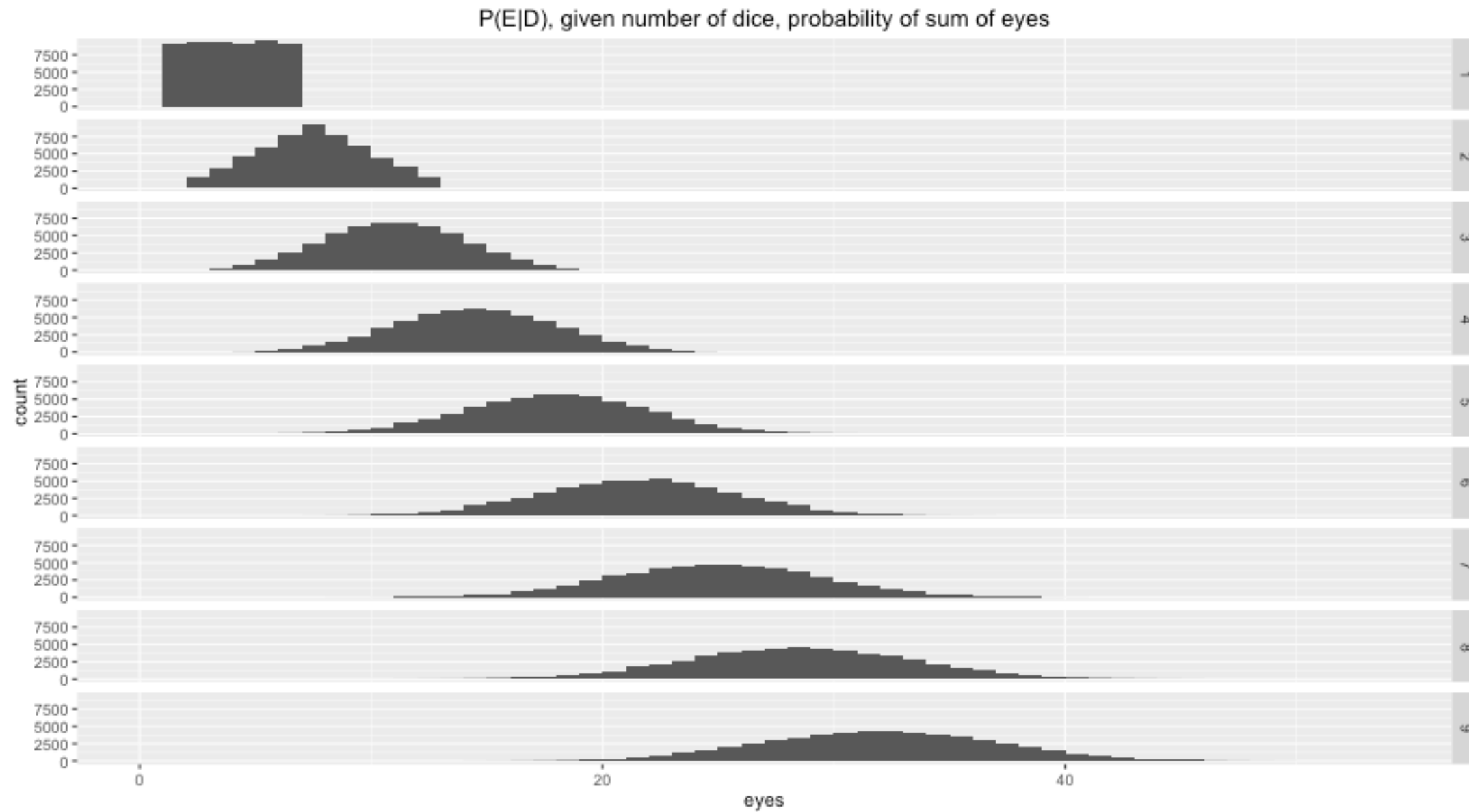
Inference via Sampling

Being able to sample allows us to not have to resort to maths.

Sometimes we know the characteristics of a system but we'd like to know the likelihood of a certain event happening. Again we can use sampling instead of maths to do the inference for us.

The next slide contains a sampling task containing dice.

Inference via Sampling



Inference via Sampling

This was the most basic example I could think of that drives the point home.

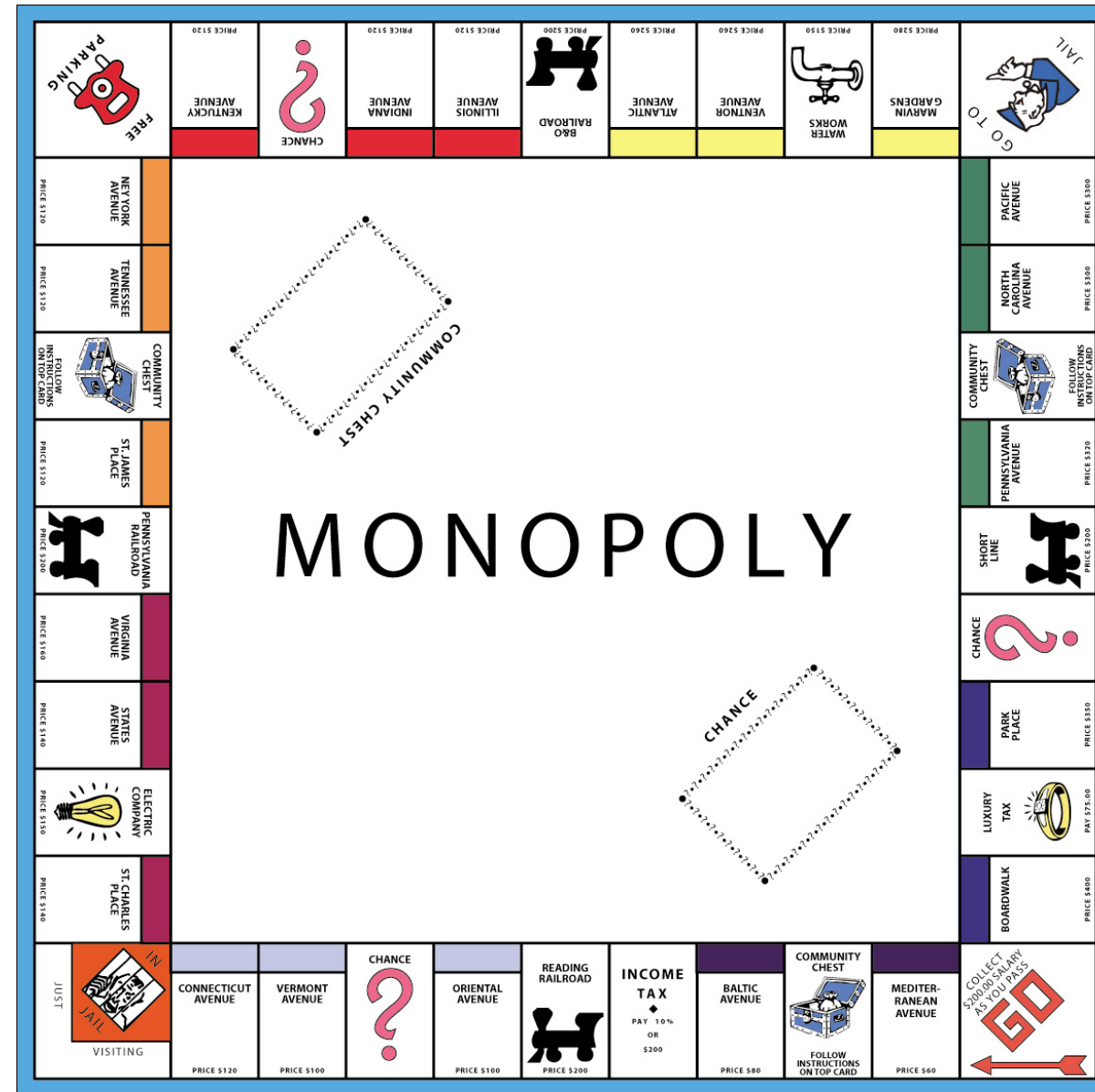
If you are interested in slightly more advanced approaches consider checking out PyMC3 or emcee.

I've also got a tutorial on a sampling algorithm for a timeseries task at my blog. It is a bit advanced but if you know scikit learn, you may learn a thing or two.

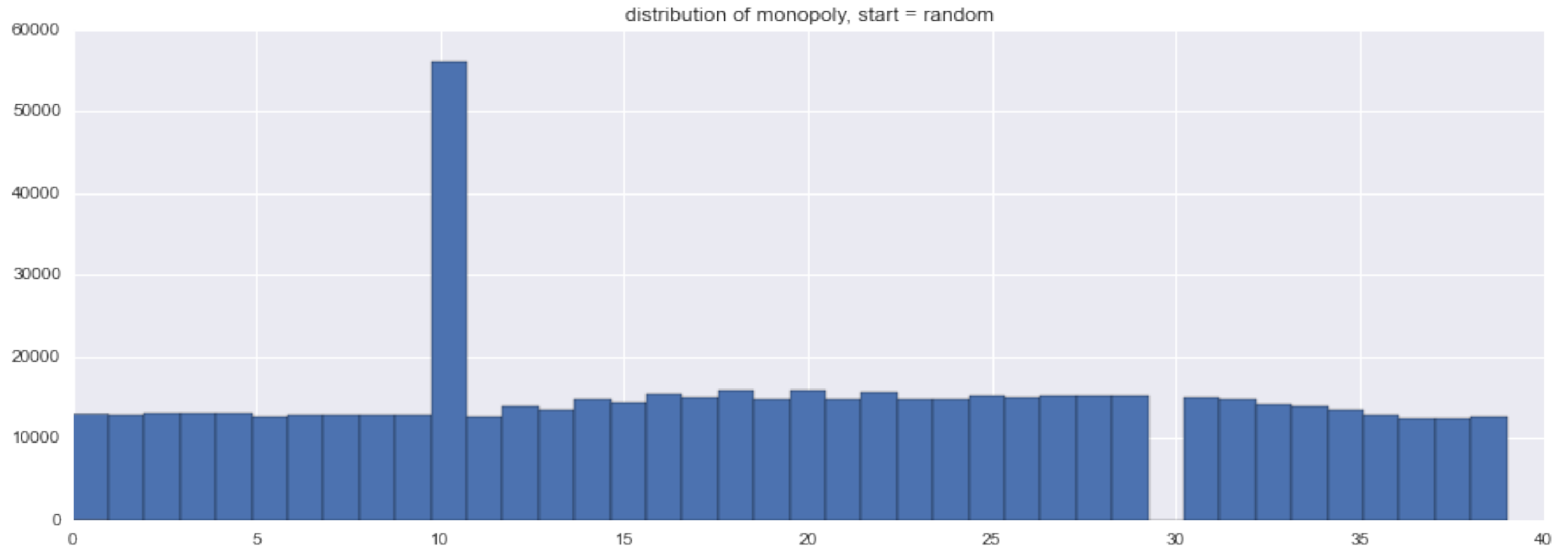
Let's consider a fun example of this

Monopoly!

Monopoly



Monopoly



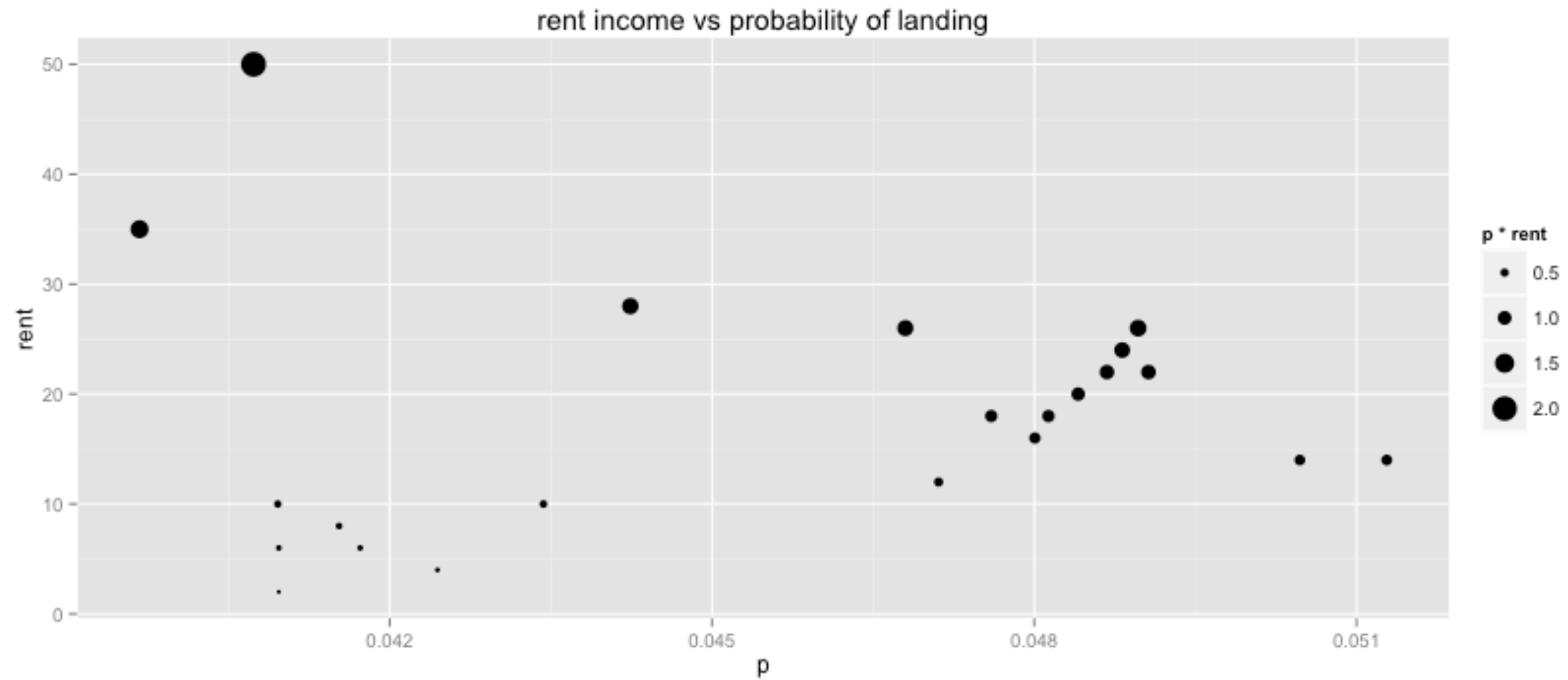
Monopoly

tile	name	prob
5	reading	0.233769
15	pennsylvania	0.255849
25	b&o railroad	0.270930
35	short line	0.239452

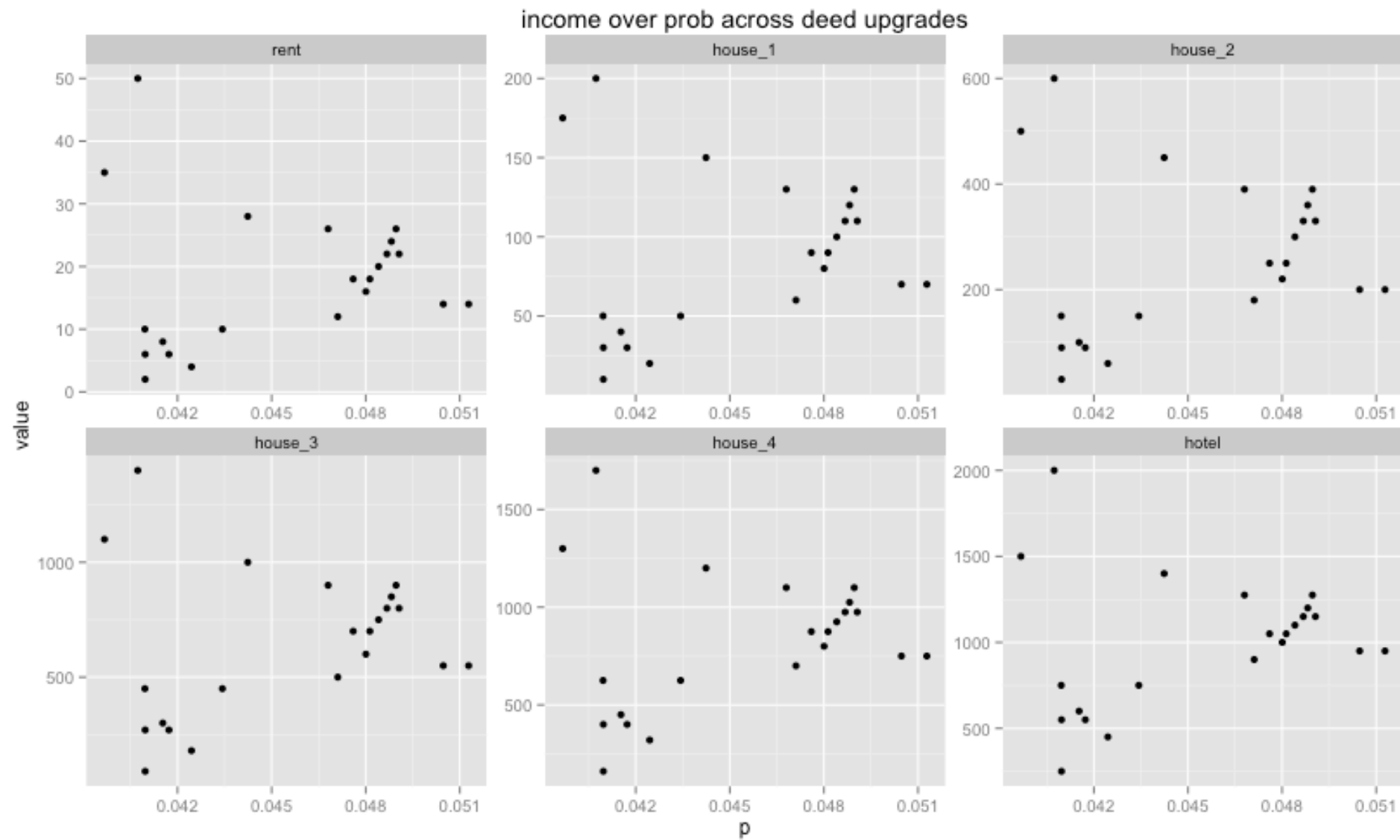
Monopoly

	name	color	p	e
1	Mediterranean Avenue	purple	0.0410	0.0819
2	Baltic Avenue	purple	0.0424	0.1698
3	Oriental Avenue	light_blue	0.0410	0.2458
4	Vermont Avenue	light_blue	0.0417	0.2504
5	Connecticut Avenue	light_blue	0.0415	0.3322
6	St. Charles Place	pink	0.0410	0.4096
7	States Avenue	pink	0.0434	0.4343
8	Virginia Avenue	pink	0.0471	0.5653
9	Tennessee Avenue	orange	0.0505	0.7066
10	St. James Place	orange	0.0513	0.7179
11	New York Avenue	orange	0.0480	0.7681
12	Kentucky Avenue	red	0.0481	0.8664
	...			

Monopoly



Monopoly



Let's consider something profitable

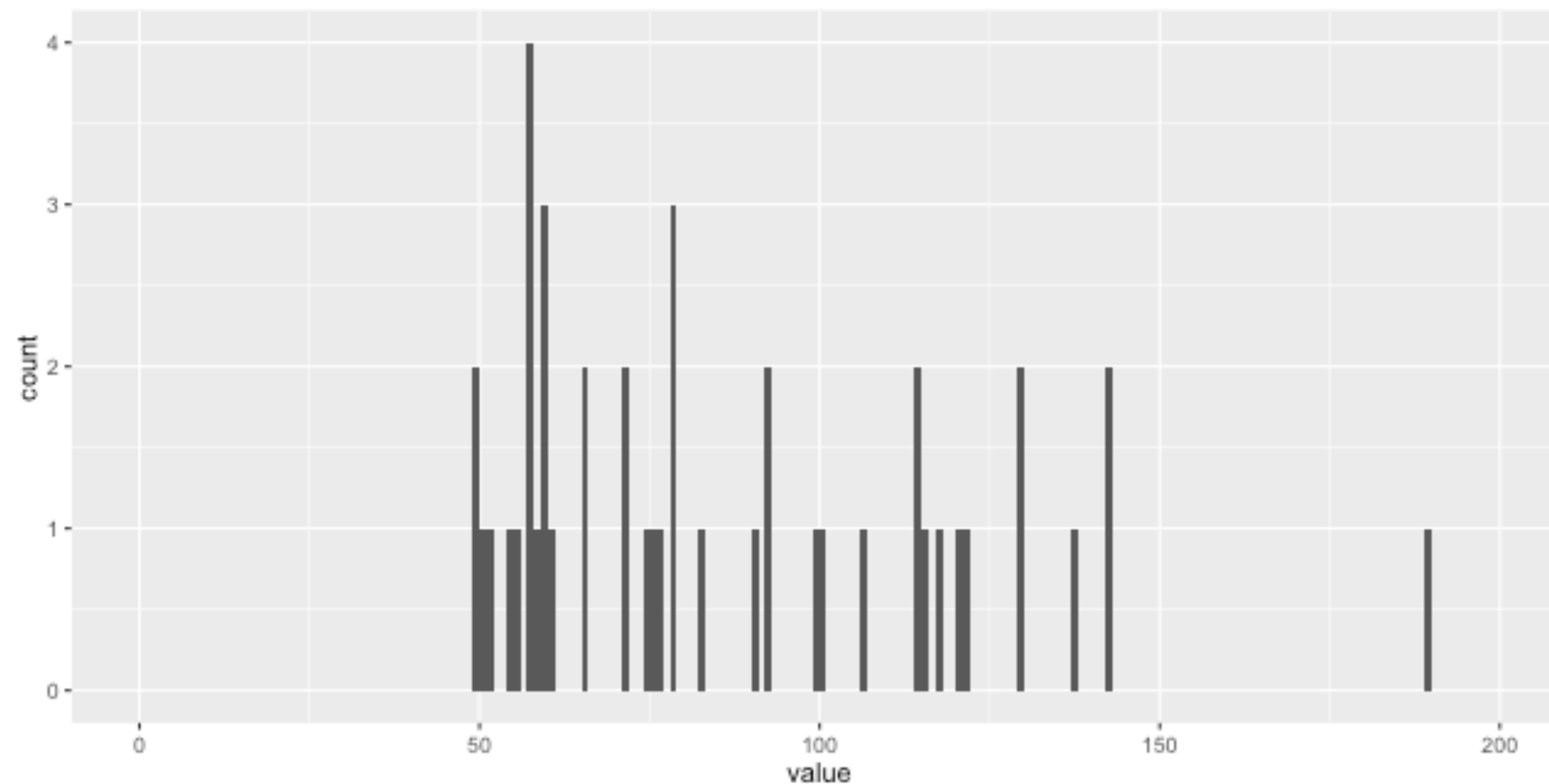
Lego Minifigures!

LEGO Minifigures



Lego Minifigures

Acquired data for Simpsons Minifigures.



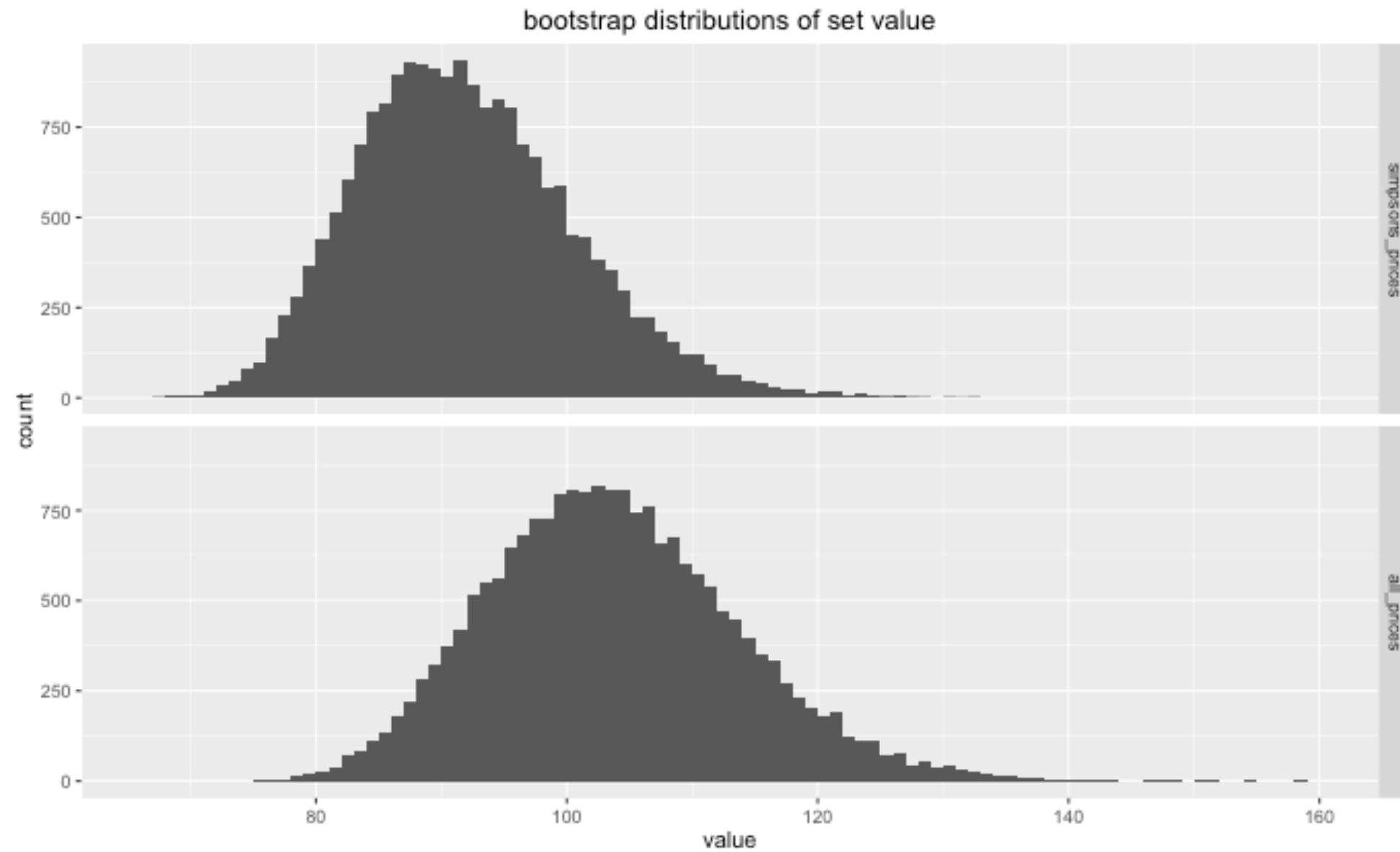
Lego Minifigures

Cleanup via sampling.

- Grab 30 prices at random
- Calculate an average
- Repeat

This should give me an image of what the mean distribution of prices might look like.

Lego Minifigures



Lego Minifigures

A minifigure costs about 3 euro a piece.

We can sell a set for 100 euro later.

How likely is it have a full set?

Math Overflow to the Rescue?

elements into k non-empty parts. If the parts are to be labelled with the days 1 to k of the year, then any division into k non-empty parts gives rise to $k!$ divisions into labelled non-empty parts. Hence the number of divisions into k labelled non-empty parts is $k!S(n, k)$. There is an inclusion/exclusion formula for $S(n, k)$ early in the article (not useful for large k) and there are recurrences later. – André Nicolas Dec 30 '15 at 21:48

[add a comment](#)

↑
1
↓
✓

Let S be the set of all assignments of birthdays to the n people, and let A_i be the set of assignments for which day i is not represented, for $1 \leq i \leq k$.

Then the number of assignments for which every day is represented is given by


$$|\bar{A}_1 \cap \dots \cap \bar{A}_k| = |S| - \sum_i |A_i| + \sum_{i < j} |A_i \cap A_j| - \sum_{i < j < k} |A_i \cap A_j \cap A_k| + \dots$$
$$= k^n - \binom{k}{1}(k-1)^n + \binom{k}{2}(k-2)^n - \binom{k}{3}(k-3)^n + \dots + (-1)^{k-1} \binom{k}{k-1} 1^n,$$

so the probability that every day is represented as a birthday is equal to

$$\frac{k^n - \binom{k}{1}(k-1)^n + \binom{k}{2}(k-2)^n - \binom{k}{3}(k-3)^n + \dots + (-1)^{k-1} \binom{k}{k-1} 1^n}{k^n}$$
$$= 1 - \binom{k}{1} \left(1 - \frac{1}{k}\right)^n + \binom{k}{2} \left(1 - \frac{2}{k}\right)^n - \binom{k}{3} \left(1 - \frac{3}{k}\right)^n + \dots + (-1)^{k-1} \binom{k}{k-1} \left(1 - \frac{k-1}{k}\right)^n$$

[share](#) [cite](#) [edit](#) [flag](#)

answered Dec 31 '15 at 0:35

 **user84413**
19.7k ● 1 ■ 12 ▲ 42

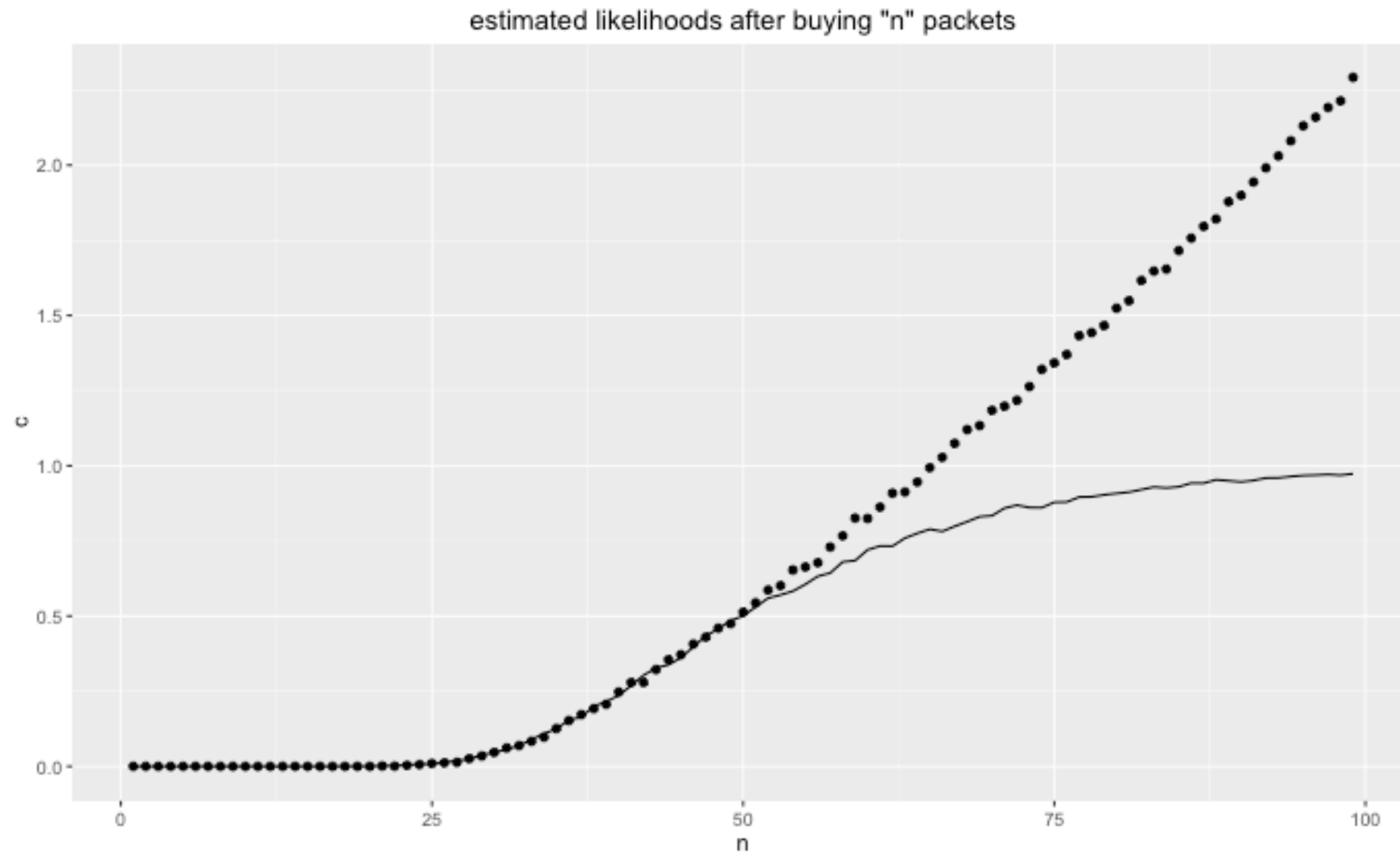
[add a comment](#)

[Answer Your Question](#)

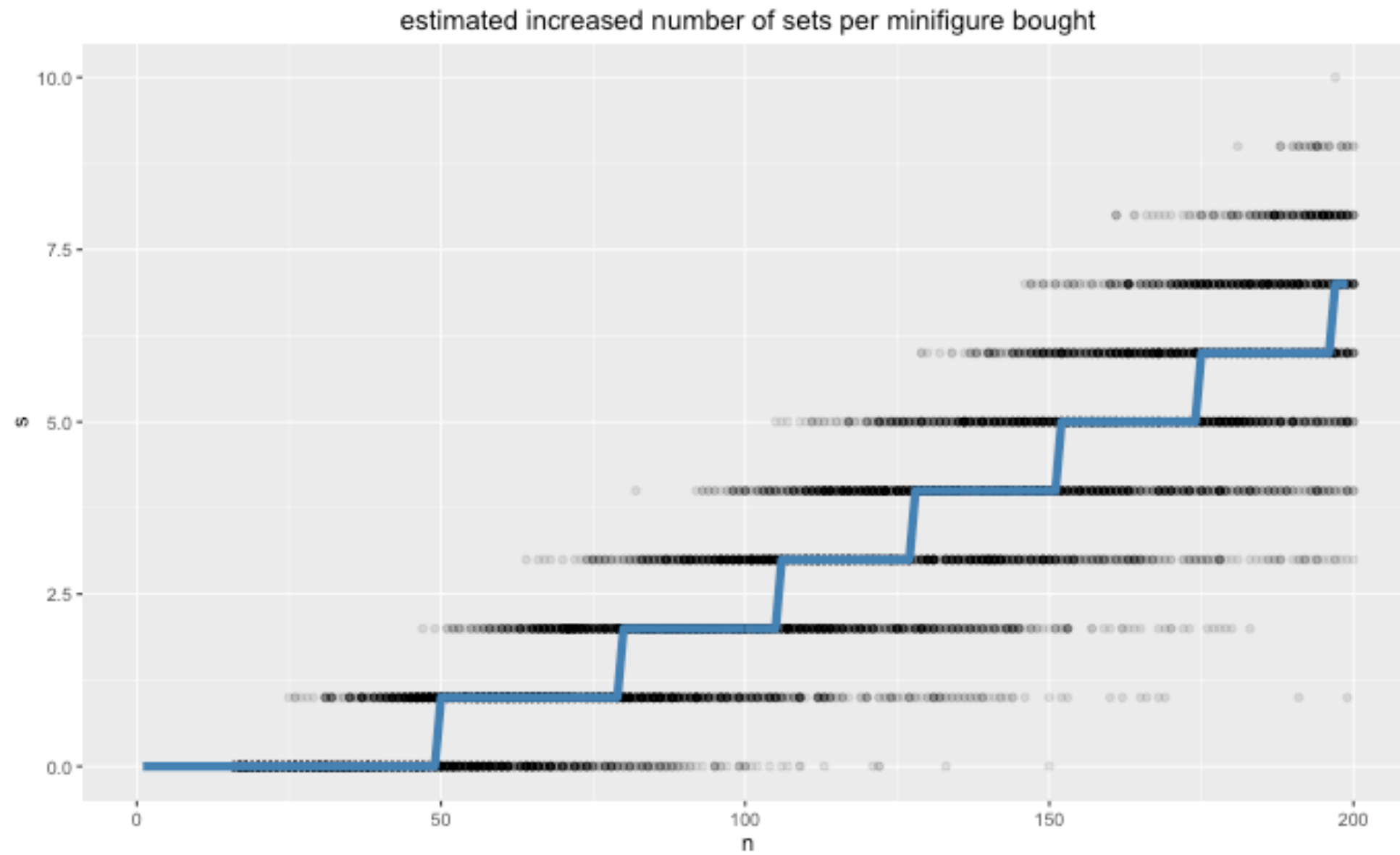
Hot

- Ca
- Po
- W
- ho
- en
- gr
- M
- rel
- mc

Lego Minifigures



Lego Minifigures



Step 3: Profit (... some day)

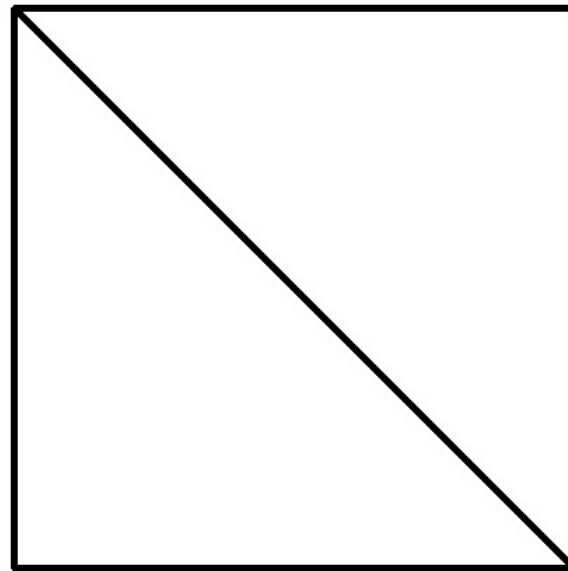


Let's talk about general usecases...

Optimisation!

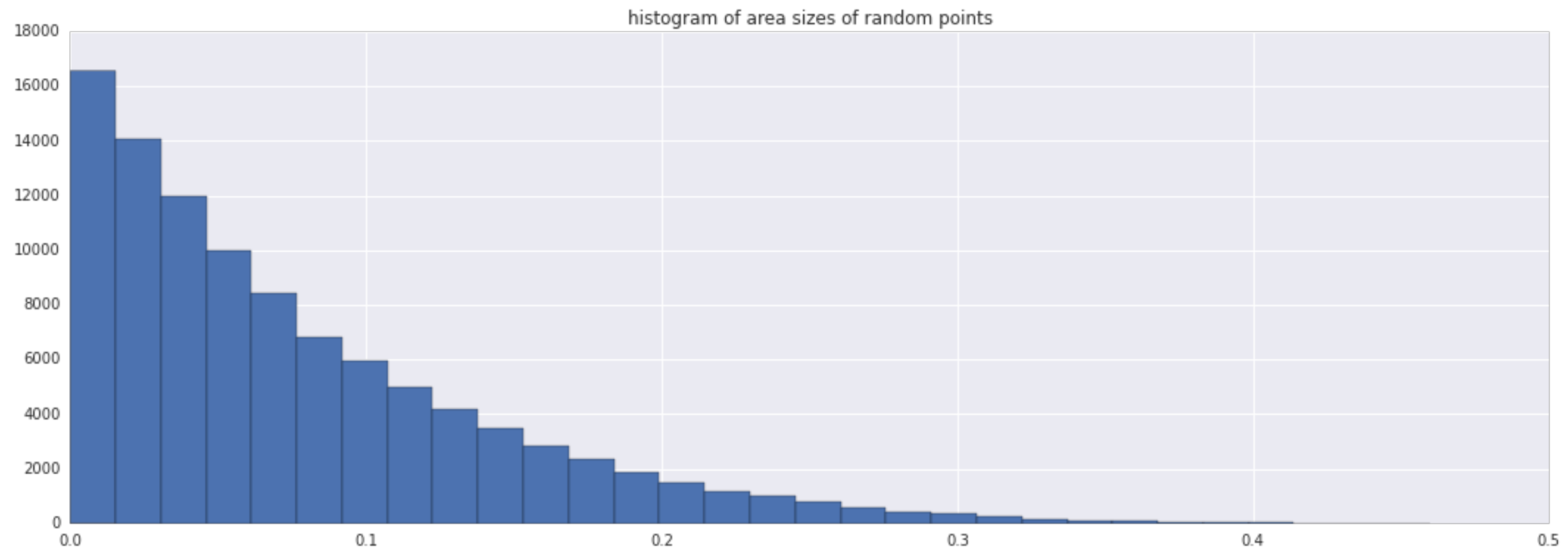
Optimisation!

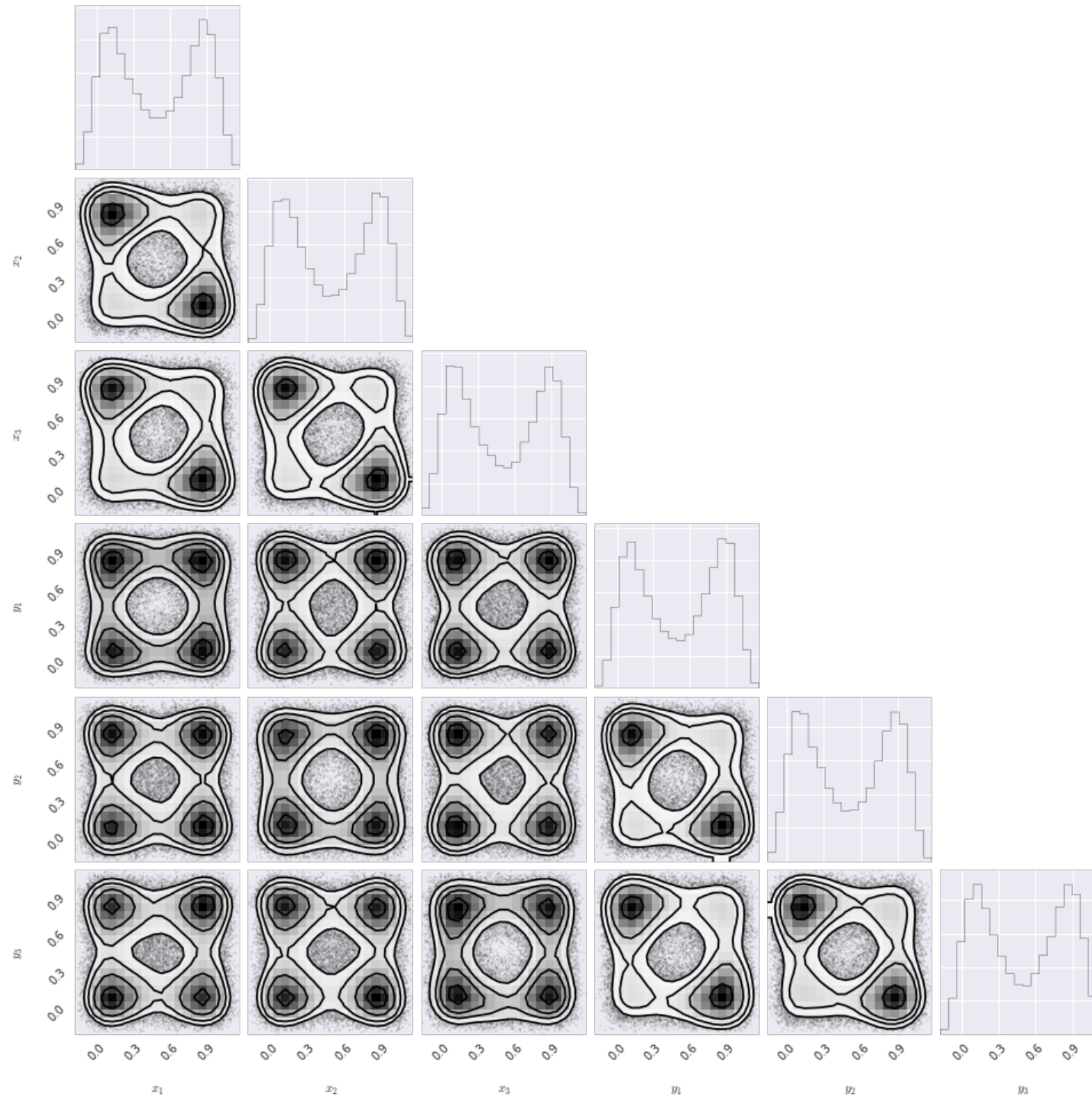
Sometimes sampling can help with an optimisation problem!
Let's take a simple example; finding the largest triangle in a
1x1 square.

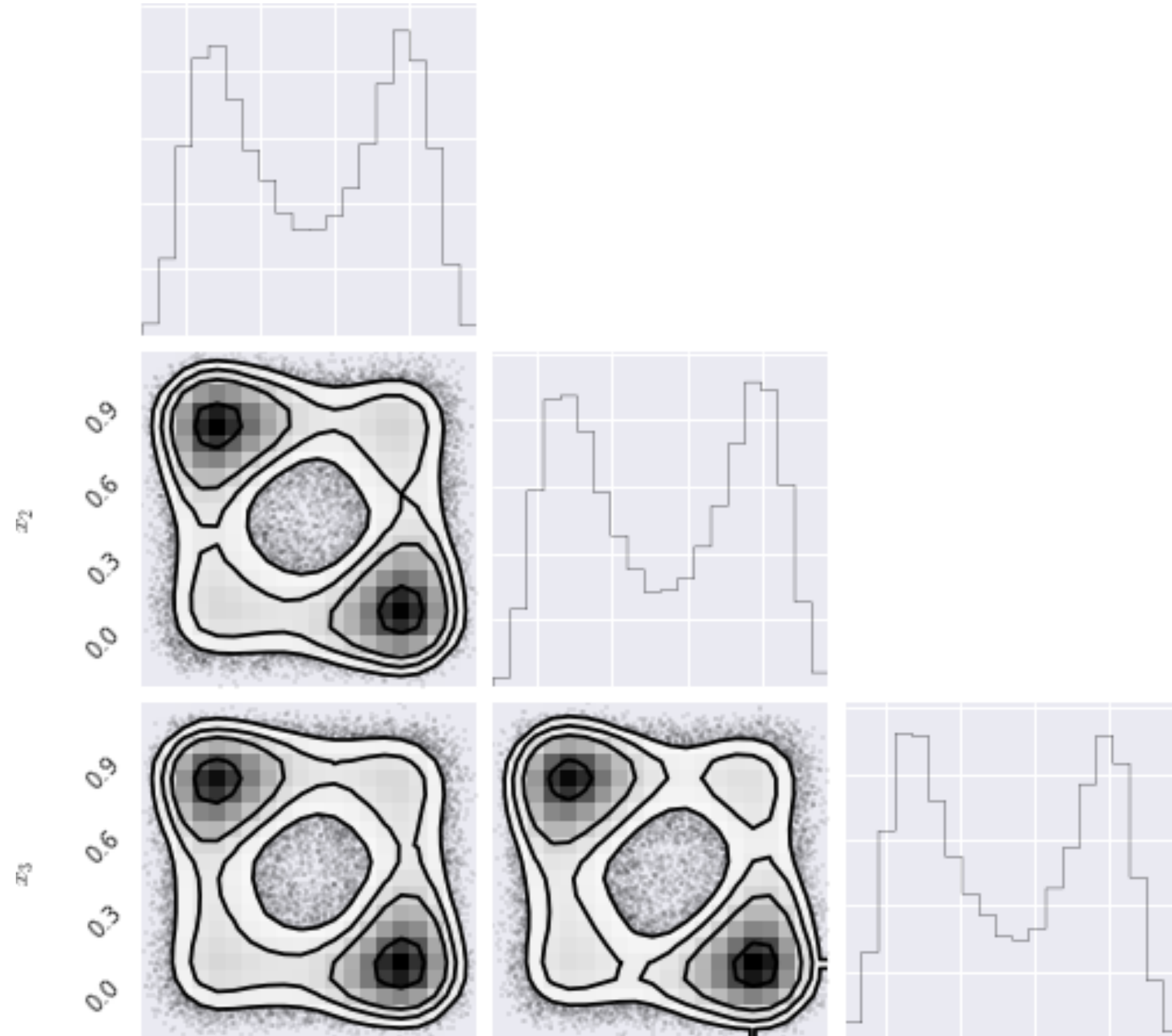


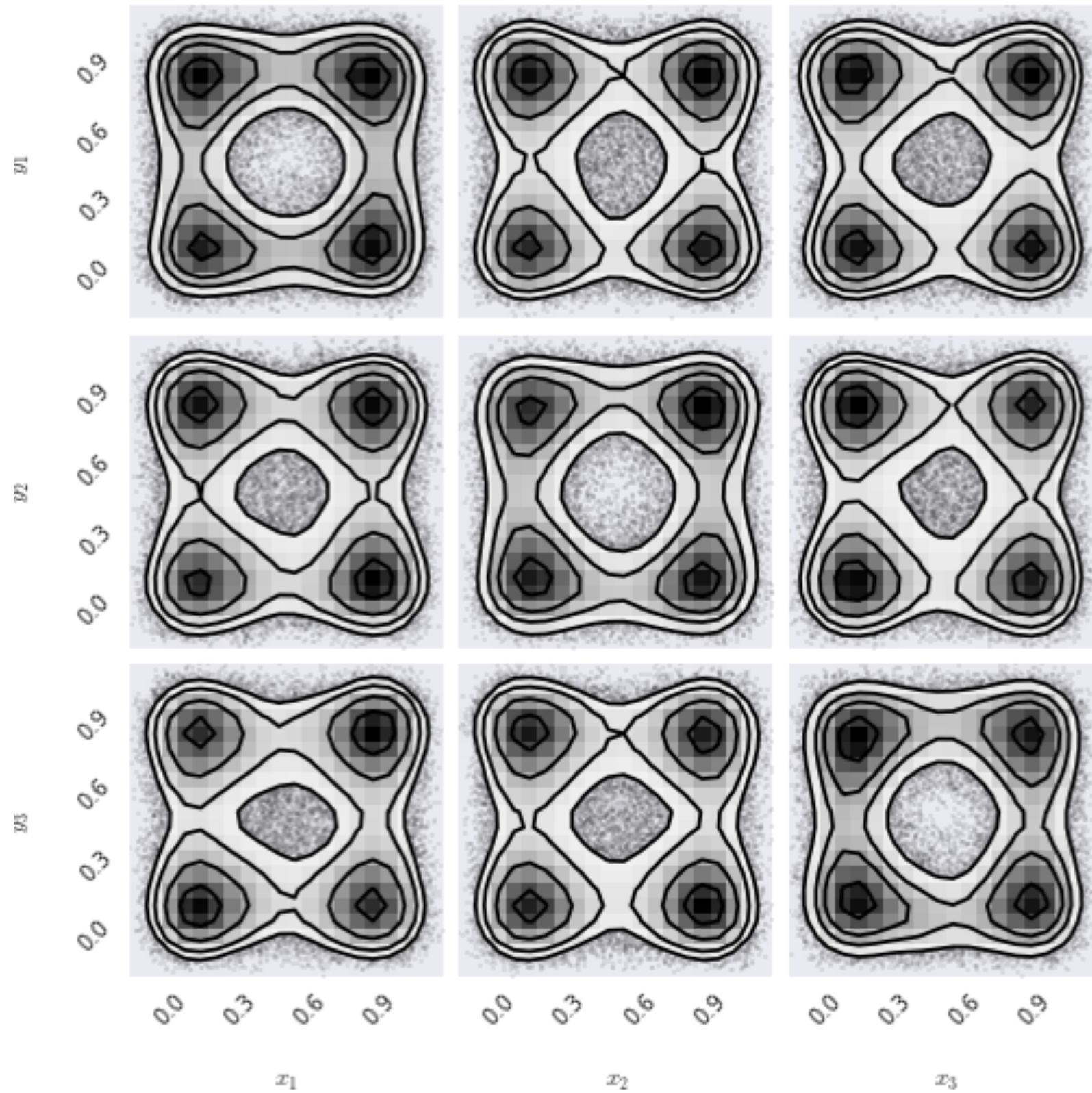
Let's start with random

```
rand_vals = np.random.rand(100000, 6)  
_ = plt.hist([shoelace(i) for i in rand_vals], 30)
```







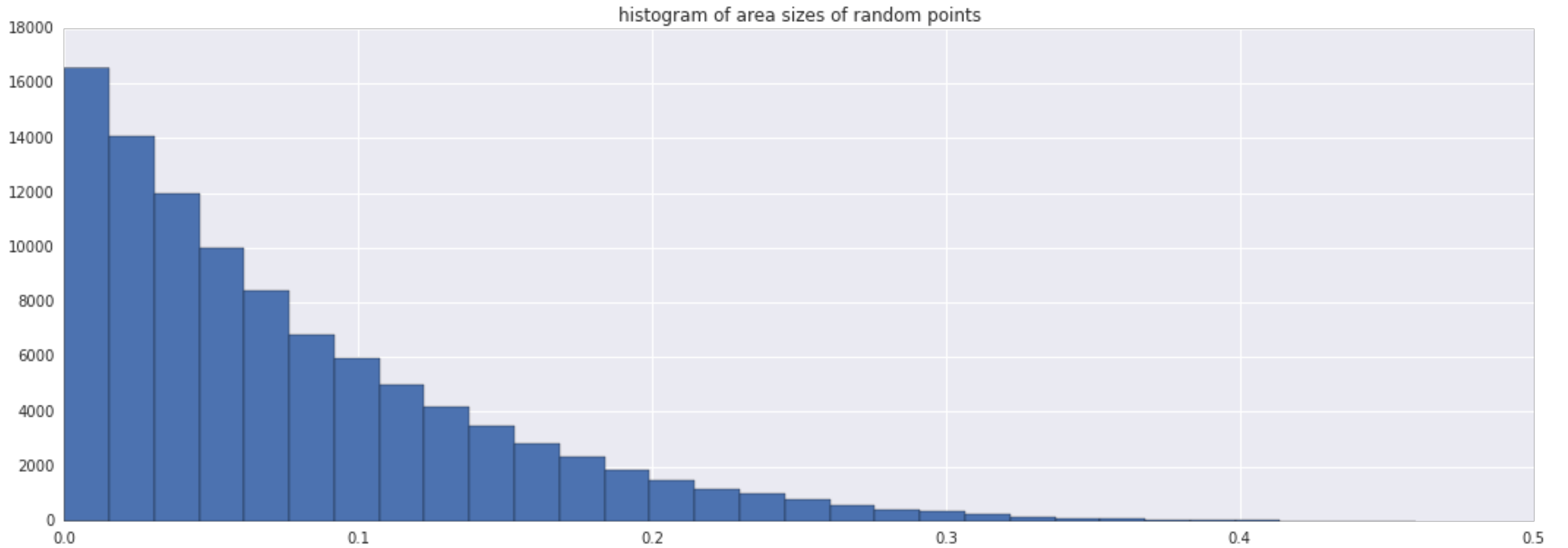


Learning step

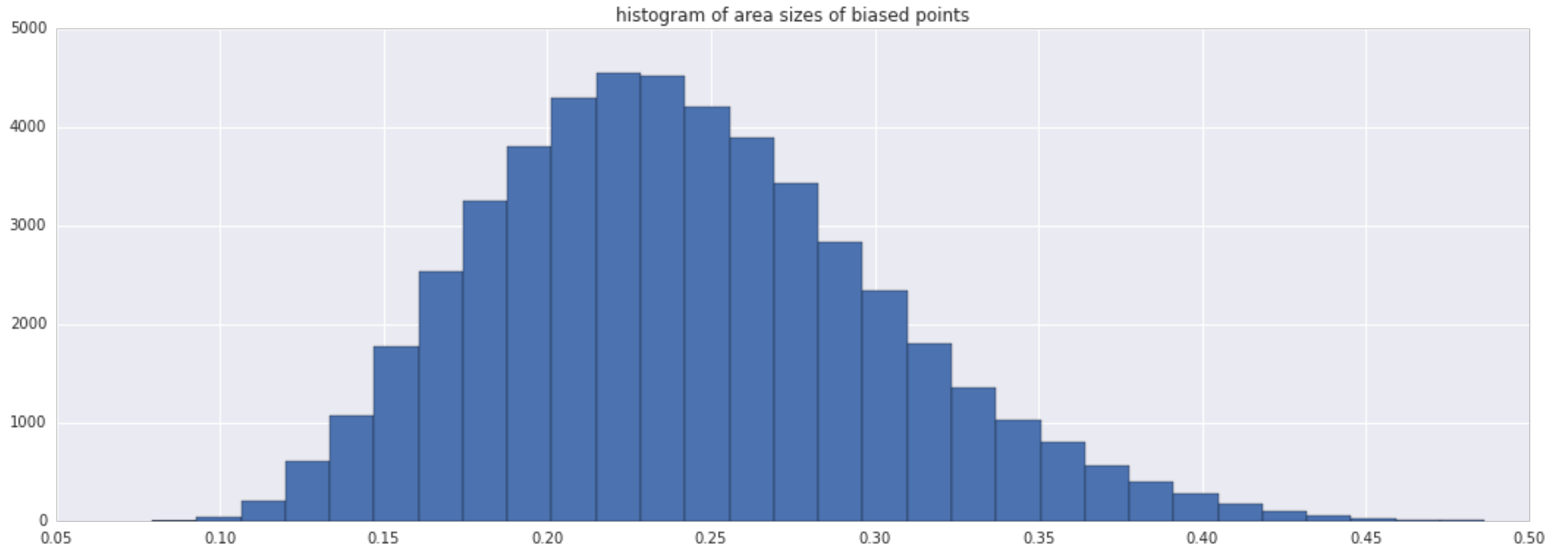
This Probabilistic approach allows us to learn many things from just sampling!

We've just shown $p(X|A > m)$, if we sample new coordinate points from this distribution, what does the new area histogram look like?

Before



After



Next steps

We can repeat the same idea until some form of convergence.

Note that m can be chosen in automatic fashions as well, ie.

$$m = \bar{A}.$$

The cool bonus with these algorithms is that we can use inference on our simulated data to learn more about the nature of our optimisation problem.

Genetic algorithms work in a very similar way!

Next steps

If you're interested in learning more about this you'll want to see my colleague's talk on this topic.

He'll talk more in detail about these sampling methods for optimisation. In this specific talk he addresses how to win at the boardgame Risk.

"How to Conquer the World"

Thursday (12:00, PyCharm room)

Things that I am most interested in.

**Generative
Methods...**

(... to outsource Creativity)

Outsourcing Creativity

Sofar distributions have been rather static. In this next bit I'll introduce a more markovian way of thinking about randomness.

Outsourcing Creativity

Does anybody understand this joke?

my linkedin profile

R, python, javascript, shiny, dplyr, purrr, ditto, ggplot, d3, canvas, spark, sawk, pyspark, sparklyR, lodash, lazy, bootstrap, jupyter, vulpix, git, flask, numpy, pandas, feebas, scikit, pgm, bayes, h2o.ai, sparkling-water, tensorflow, keras, onyx, ekans, hadoop, scala, unity, metapod, gc, c#/c++, krebase, neo4j, hadoop.

Outsourcing Creativity

Recruiters cannot distinguish a pokemon name vs. a name of a technology.

I figured making a python library that can generate pokemon names might be fun (**grabble** = tech names as a service).

...

Outsourcing Creativity

Recruiters cannot distinguish a pokemon name vs. a name of a technology.

I figured making a python library that can generate pokemon names might be fun (**grabble** = tech names as a service).

... speaking of pokemon as tech names.

Repokémon

Showcase of GitHub repos with Pokémon names

★ Star 56 🍴 Fork 6 🐦 Tweet 👍 Like Share 4



Bulbasaur
Helper Preadly
crawler operations
★2 🍴2



Ivysaur
Generate options
for specified tickets
★0 🍴0



Venusaur
landscaping
prototype
★0 🍴0



Charmander
Charmander
Scheduler Lab -
Mesos, Docker,
InfluxDB, Spark
★38 🍴11



Charmeleon
Chokidar wrapper
to avoid
Segmentation
faults (named after
a joke by ...
★2 🍴0



Charizard
Yet another simple
php router
★10 🍴2



Squirtle
Squirtle the
Browser-based
NTLM Attack
Toolkit
★4 🍴3



Wartortle
A Pokemon MMO
Emulator
★0 🍴0



Blastoise
tiny relational
database
★11 🍴1



Caterpie
★0 🍴0



Metapod
A template-based
robot dynamics
library
★9 🍴8



Butterfree
Colocación
Idiomas
★0 🍴0



Weedle
★0 🍴0



Kakuna
ES6 promise
wrapper around
superagent.
★7 🍴0



Beedrill
Big Black Book
★0 🍴0



Pidgey
SMTP as a Service
★2 🍴0



Pidgeotto
An speech
example: message
exchange using
different
approaches (HTTP,
...
★1 🍴0



Pidgeot
★2 🍴0



Rattata
Scala Abstract
Syntax Tree(AST)
pretty print tool
★6 🍴0



Raticate
FIAT
★0 🍴0



Spearow



Fearow



Ekans
Ekans - Game Jam
2012
★2 🍴1



Arbok
★0 🍴0



Pikachu
Pikachu made with
HTML+CSS3
★4 🍴0



Raichu
★0 🍴0



Sandshrew
HTTP and SPDY
proxy written in
Java using Netty
★1 🍴0



Sandslash



Nidoran♀



Nidorina
Brito
★0 🍴0

Outsourcing Creativity

Recruiters cannot distinguish a pokemon name vs. a name of a technology.

I figured making a python library that can generate pokemon names might be fun (**grabble** = tech names as a service).

Never wrote a library before and the problem seemed interesting enough. I will most likely learn from doing this.

It will probably result in a twitter bot some day.

Short Term Plan

The whole point of grabble, is to come up with a better name.

Outsourcing Creativity

The general problem of this hobby project involves generating believable sequences of tokens.

Things like:

- pokemon names

Outsourcing Creativity

The general problem of this hobby project involves generating believable sequences of tokens.

Things like:

- pokemon names
- red hot chilli pepper lyrics

Outsourcing Creativity

The general problem of this hobby project involves generating believable sequences of tokens.

Things like:

- pokemon names
- red hot chilli pepper lyrics
- ikea furniture names

Outsourcing Creativity

The general problem of this hobby project involves generating believable sequences of tokens.

Things like:

- pokemon names
- red hot chilli pepper lyrics
- ikea furniture names
- notes on a piano

Outsourcing Creativity

Simplest model:

$$p(t_{i+1} | t_i)$$

For every pair of tokens, keep track how often they occur together. Once you have a start token, you now have a bag of words with probabilities to sample from.

Outsourcing Creativity

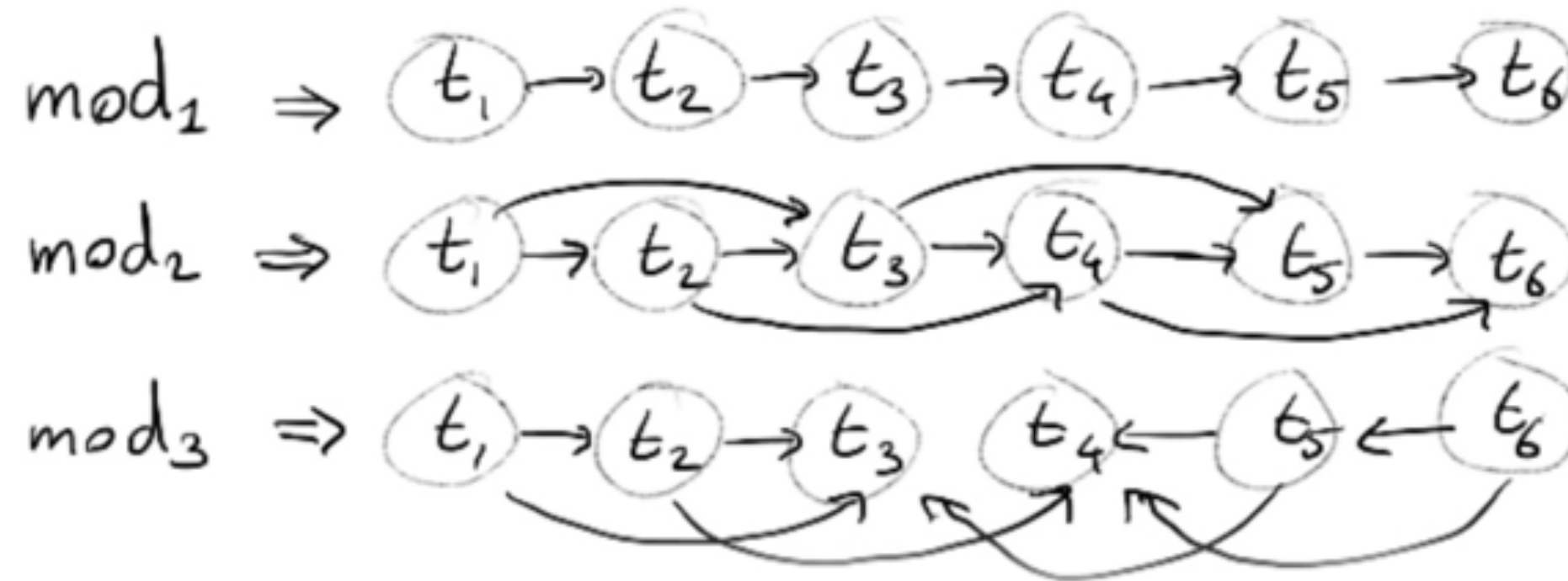
More complex model:

$$p(t_{i+1} | t_i, t_{i-1})$$

Do the same thing but for three tokens.

Outsourcing Creativity

Models that I'm considering:



Outsourcing Creativity

Some pokemon results:

lydo

keen

wqool

ryrys

poole

utcala

youtail

olma

eltyp

Outsourcing Creativity

Some red hot chilli pepper results:

Can you believe. Hold me please.

By the way I wonder what the wave meant.

White heat is screaming in the nearest bin.

When I was fortunate I know you must be fat this year.

And eat the sun and a Bottom Dollar.

Fox hole love Pie in your house

now let me spin Feather light but you cant move this

Outsourcing Creativity

Some ikea results:

anapa

frodok

pasig

ripe

latrank

vis

gsoo

yirbs

ilosseln

Outsourcing Creativity

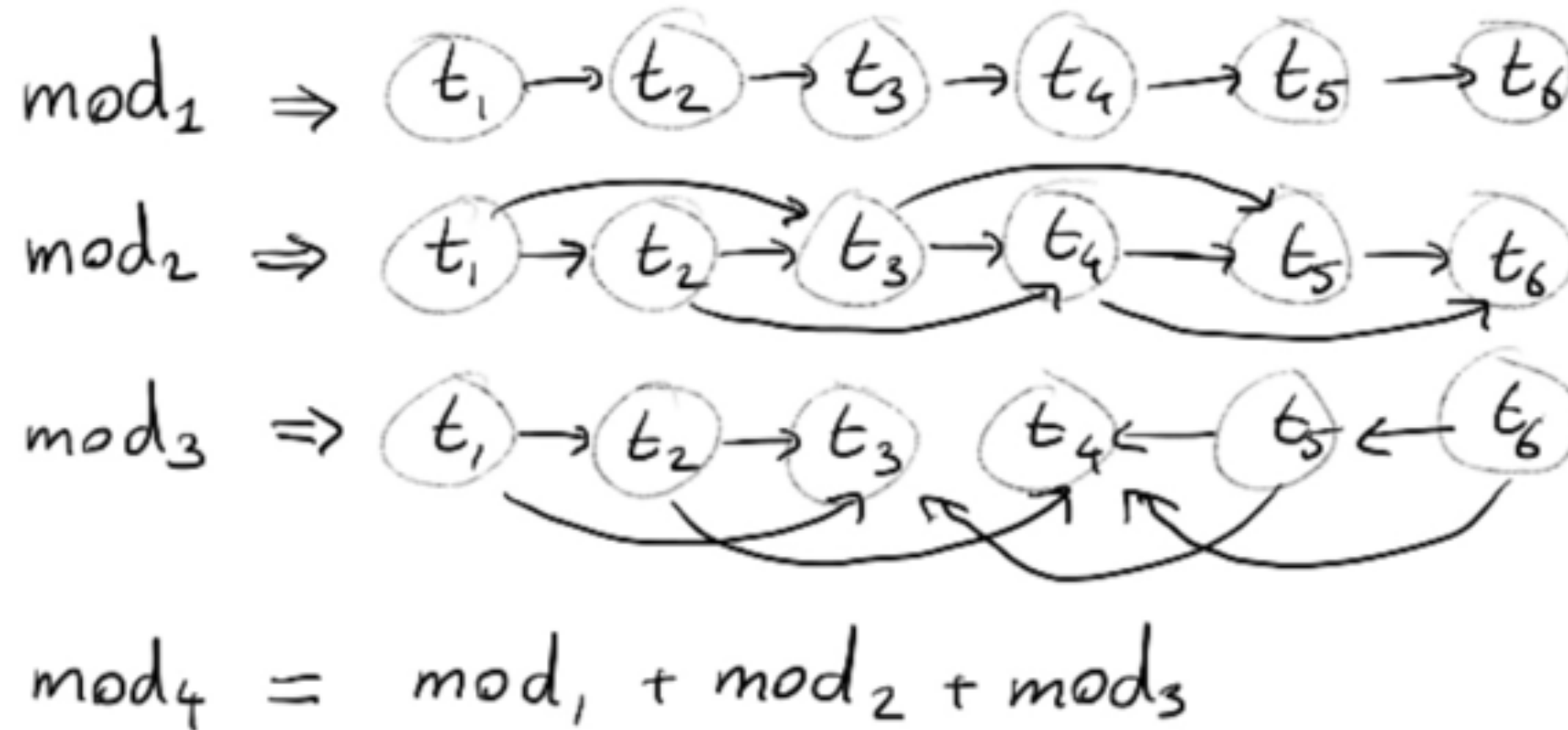
Alas, I've only shown you the nice bits.

Most of them are not good.

alrb	eaota	lavi	aepm
ajkge	oarterv	lygea	aavra
ctpfi	ilosseln	mmbor	a
dittgk	mlale	glg	stegnae
ogae	aak	niaey	lgaon
hrjke	eaalb	btomd	raiun
lrs	jrta	nntt	day
rsiht	idll	eaeann	ileaeo
redaate	dsfkvkr	rprnh	rbku
ooerngj	emee	aiaaaa	ioapu

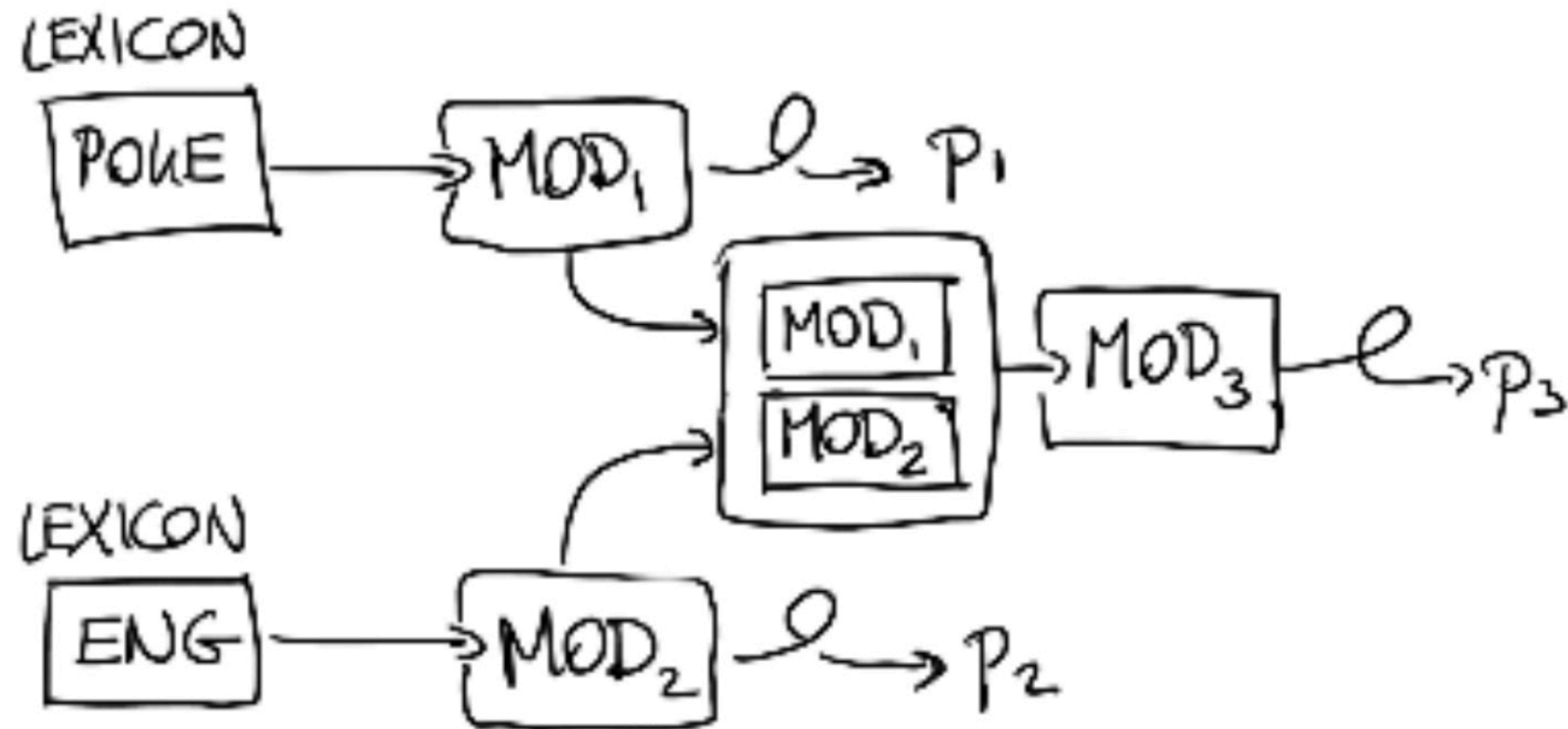
Outsourcing Creativity

One solution; make ensembles.



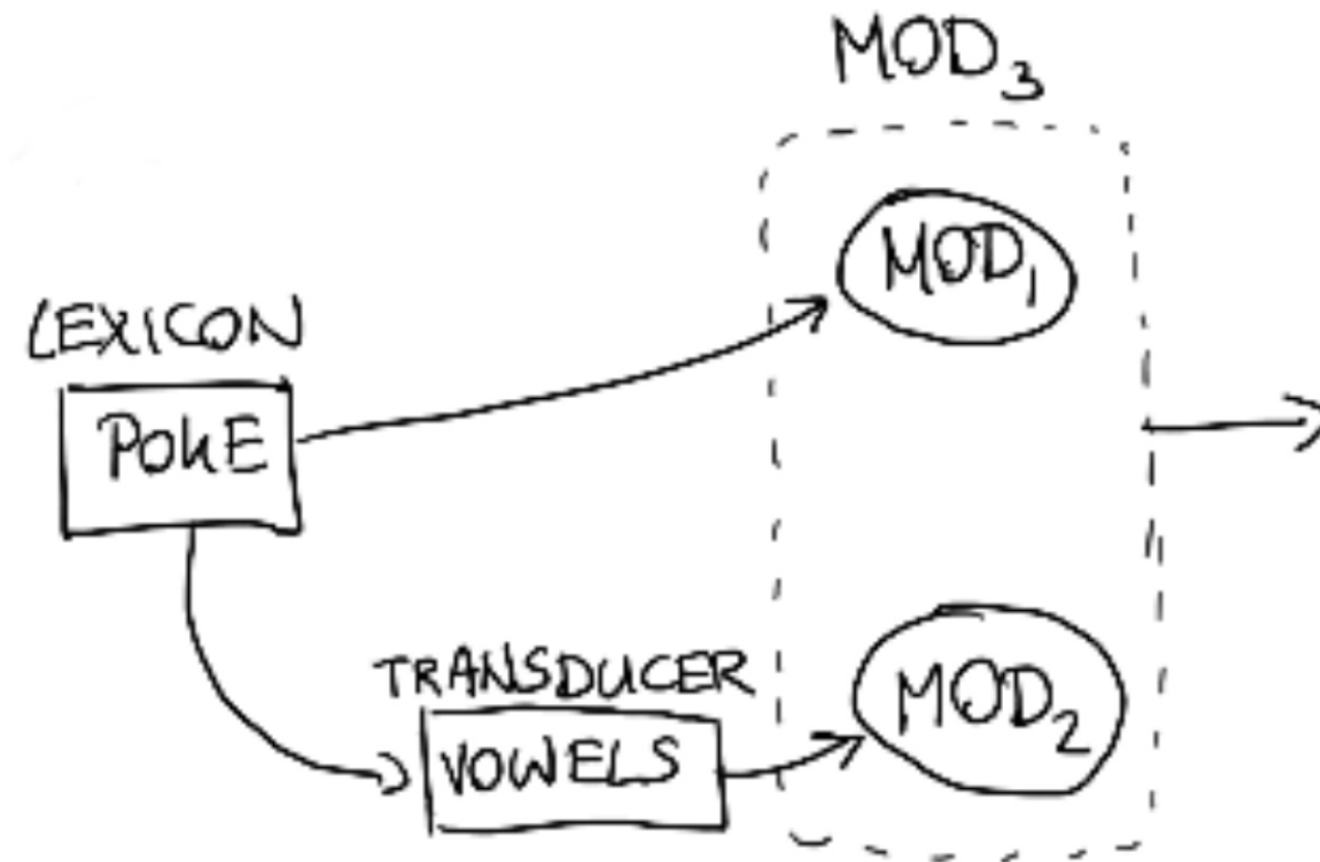
Outsourcing Creativity

One solution; combine lexicons via different models.



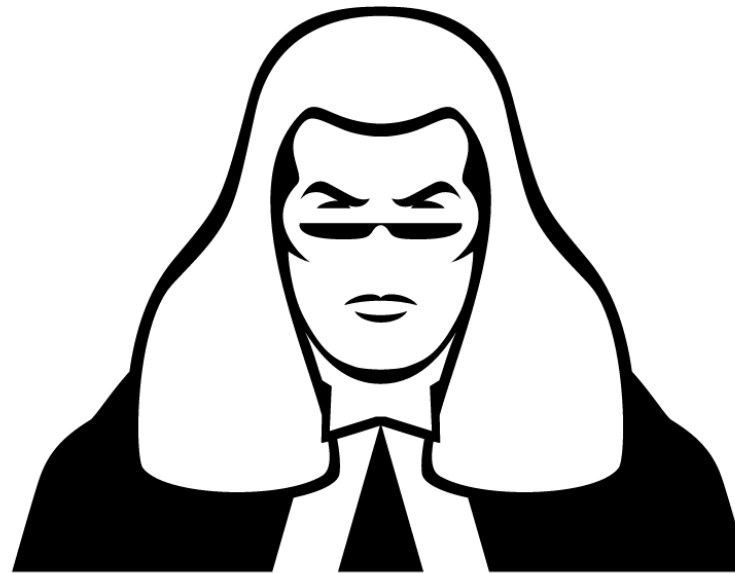
Outsourcing Creativity

One solution; add transcribers.



Outsourcing Creativity

Another solution; add judges.

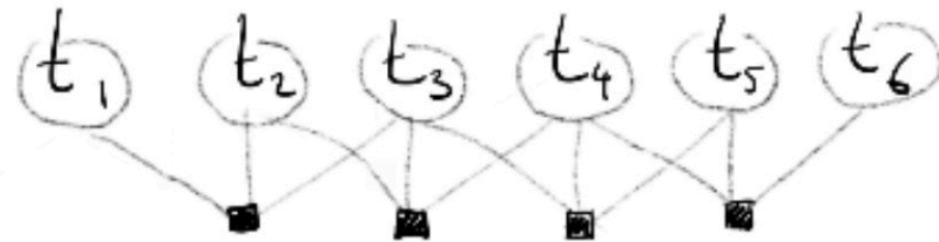


After sampling 100 samples. Maybe have another model look at the results and pick the top 10.

Outsourcing Creativity

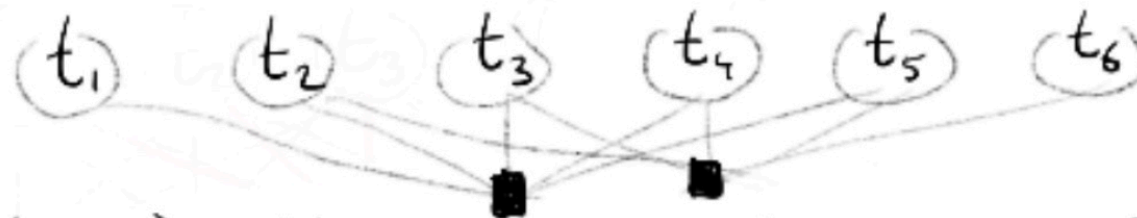
One solution; factor graph models?

FACTOR GRAPH RANK 2



$$p(t_1 \dots t_6) = f(t_1, t_2, t_3) f(t_2, t_3, t_4) f(t_3, t_4, t_5) f(t_4, t_5, t_6)$$

FACTOR GRAPH RANK 3



$$p(t_1 \dots t_6) = f(t_1, t_2, t_3, t_4, t_5) f(t_2, t_3, t_4, t_5, t_6)$$

Outsourcing Creativity

One solution; heuristics?

Levenstein-ish
approach

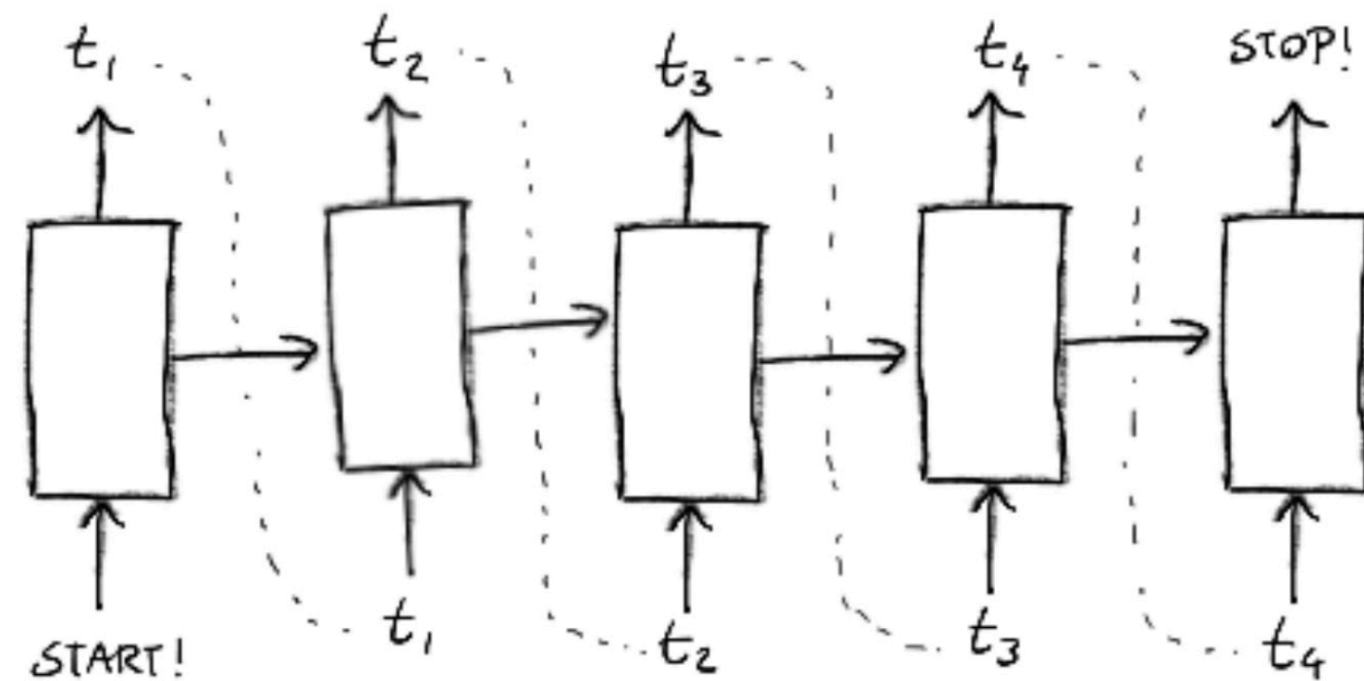
We are free to use
whatever "levenstein"-
sampling we wish.

Gensim → word2vec?
token2vec?

h	o	o	p	a	idx = 0 sample = "r"
r	o	o	p	a	
r	o	o	p	o	idx = 4 sample = "o"
r	o	e	p	o	
b	o	e	p	o	idx = 2 sample = "e"
b	o	e	h	o	
					idx = 0 sample = "b"
					idx = 3 sample = "h"

Outsourcing Creativity

One solution; deep models?

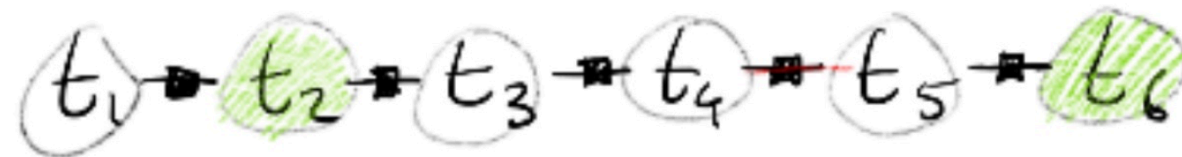


$t_1 \rightarrow t_2 \rightarrow t_3 \rightarrow t_4$

Outsourcing Creativity

Deep models have issues.

Now suppose



Tactic: t_2, t_6 are known. Next we need to sample $\{t_1, t_3, t_5\}$. These are simple to draw given previous exercise.

Next we'll need to draw t_4 . Again. This is little more than a table lookup.

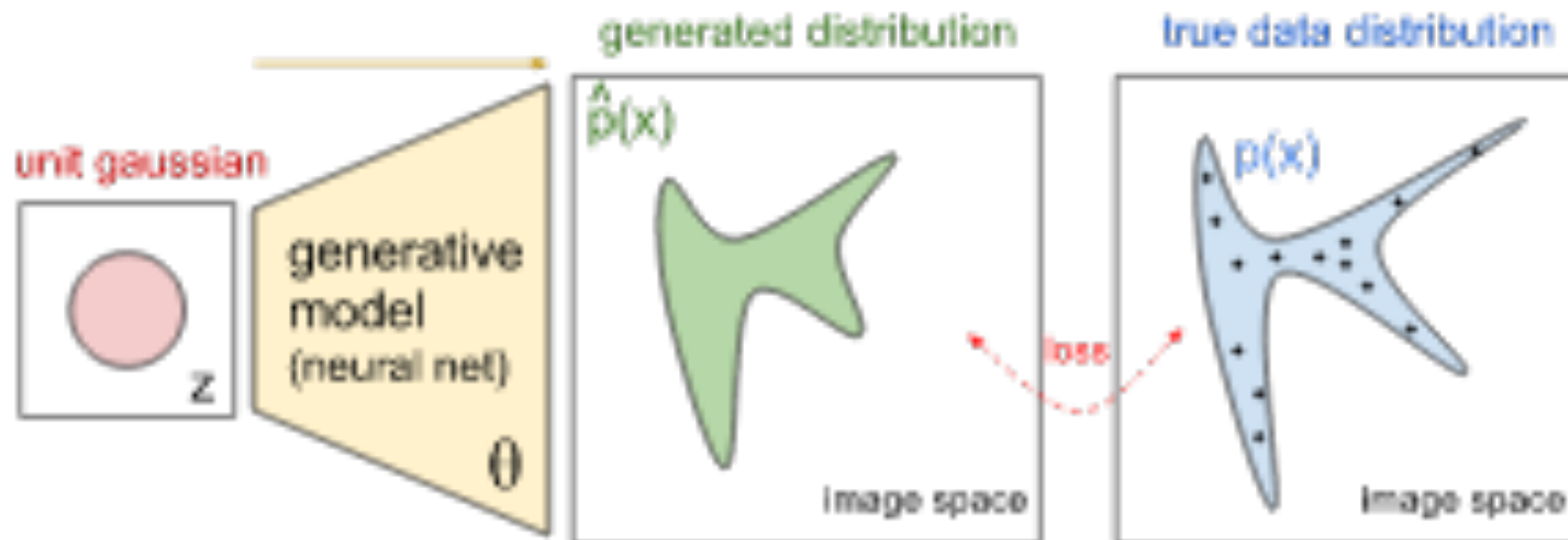
Outsourcing Creativity

I will focus on the following three algorithm domains:

- probabilistic graphical approaches
- heuristic approaches
- deep neural network approaches

Outsourcing Creativity

There's similar work being done in this field for images.



Source: [openai](https://openai.com).

Rough API Plan

```
lx1 = gb.Lexicon(filepath/iterable)
lx2 = gb.Lexicon(filepath/iterable)

mod1 = gb.Model.OneWayMarkov(n = 2, smoothing = 0.001).fit(lx1)
mod2 = gb.Model.TwoWayMarkov(n = 3)\
      .translate(gb.Transcribe.Vowel).fit(lx1)
mod3 = gb.Model.FactorGraph().fit(lx2)

mod1.generate(20)
mod2.generate(20)

ens = gb.Model.Ensemble([mod1, mod2])
ens.generate(100).sort(mod3.judge)
```

Dreams

```
mod4 = mod1.addTransducer(gb.Vowel).fit(1x1)
```

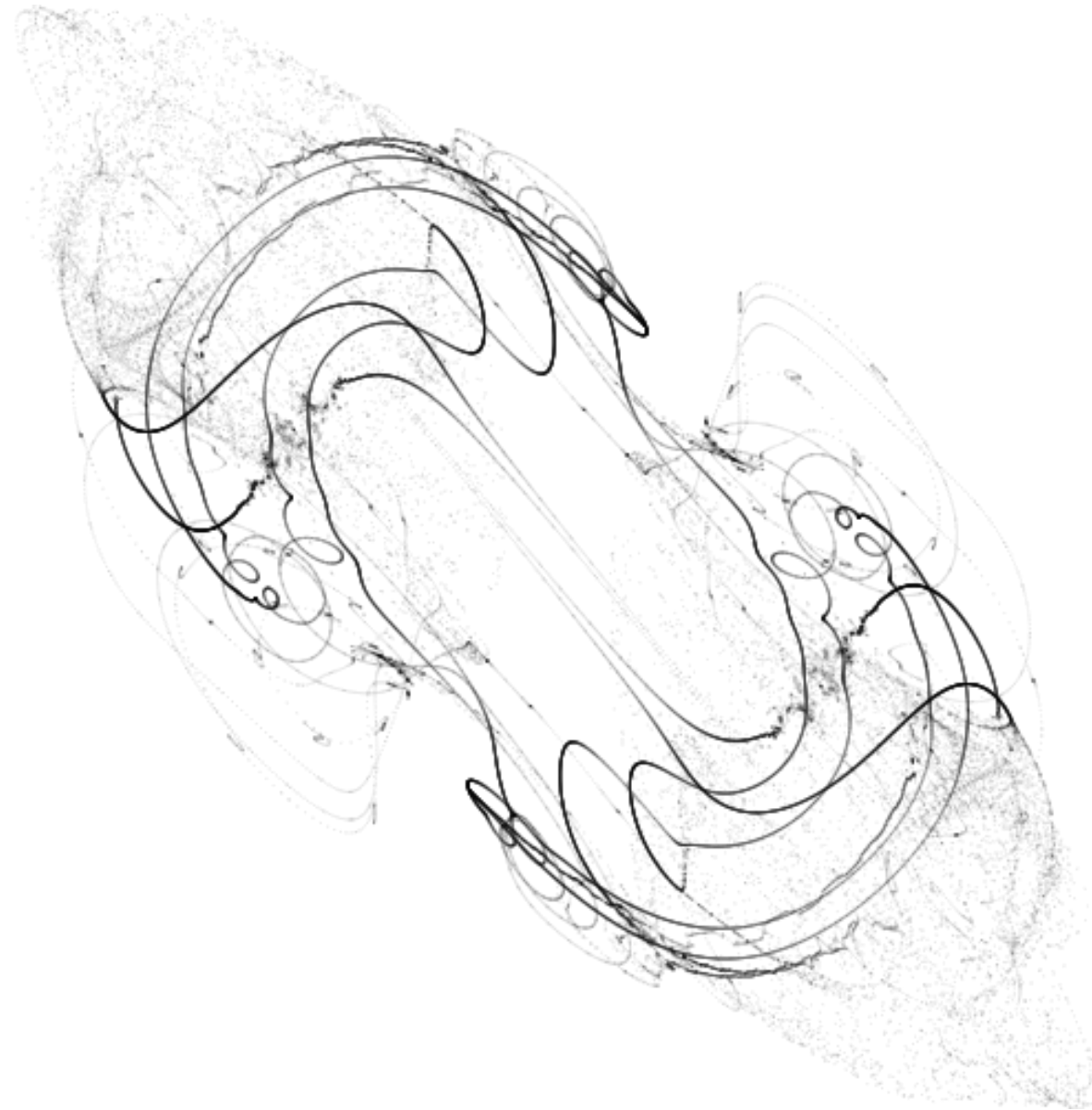
```
ens = gb.Model.Ensemble([mod1, mod2])
```

```
ens.finish(['*', '*', '*', 'B', 'A', 'S', 'E'], n = 100)
```

```
ens.finish(['*', '*', '*', 'D', 'A', 'T', 'A'], n = 100)
```

```
ens.finish(['H', 'A', '*', '*', '*', 'P', '*'], n = 100)
```

Creativity? Come see blender!



Conclusion

Sampling can be a whole lot of fun and sometimes even profitable. Getting started is easy and you might be surprised how often it can help you out.

Python is an amazing language for this usecase too.

Think about API from user first. Optimize Joy!

Shoutout to numpy, cytoolz and generators. Helpful!

Thanks for Listening!

