Higher-Order Probabilistic Programming and Name Generation

Marcin Sabok[1], Sam Staton[2], **Dario Stein[2]**, Michael Wolman[1] @damast93

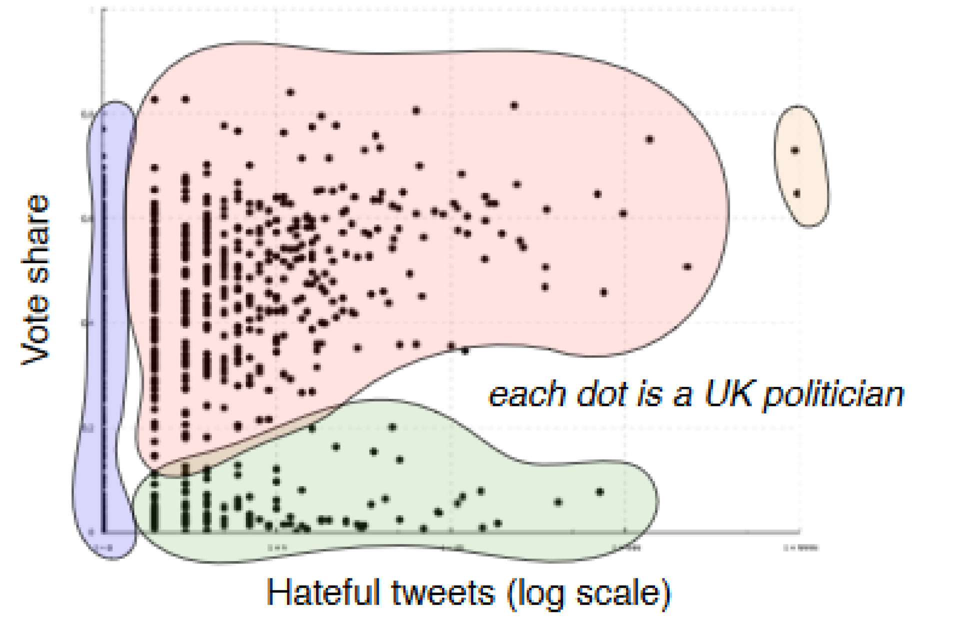
1 McGill University 2 University of Oxford

POPL'21; https://arxiv.org/abs/2007.08638

Name generation

- *Names* are unique identifiers like GUIDs, memory locations, cluster names
- `gensym` creates a fresh name
- Often treated as a probability distribution,
  e.g. as a prior for a Dirichlet process [e.g. Roy & al.'08]

```
(define draw-class (DPmem 1.0 gensym))
(define class
  (mem (lambda (obj) (draw-class))))
```



[taken from ongoing research by Hale, Kammar, Margetts, Melham, Staton, Szymczak, Vidgen]

- Can we use **random samples as fresh names**?

✓ Continuous samples have 0 collision probability

– Subtlety: Names can stay **private** inside functions

– Example: the following function `name -> bool` always return false

```
let x = gensym() in fun y -> x == y                (A*)
```

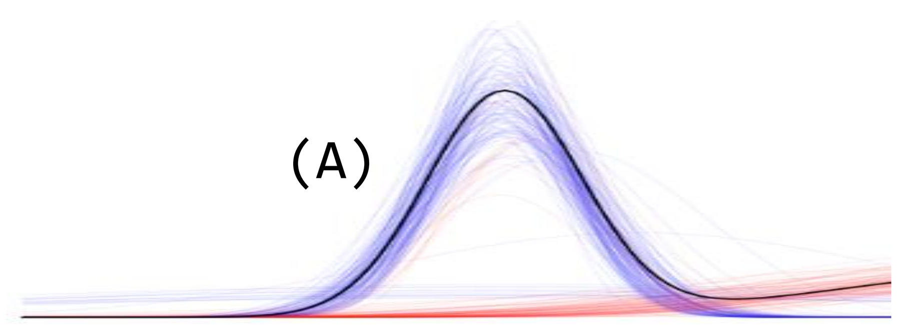
A theory of random functions

- Can the following **random function** `real -> bool`

```
let x = normal(0,1) in fun y -> x == y             (A)
```

always be replaced by the constant one?

```
fun y -> false                                     (B)
```



- **Yes:** We prove a `privacy equation' – randomizing a value anonymizes it (`x` is private)
- We cannot condition on functions being equal, as this would distinguish `(A)` and `(B)`

[Denotational Semantics: Quasi-Borel spaces]

Which mathematical framework can analyze higher-order functions + continuous distributions?

- Measure theory is insufficient [Aumann'61]; quasi-Borel spaces [Heunen'17] are a convenient tool
- Ours is the first work to analyze quasi-Borel function spaces in detail.
- *Proof sketch:* Measurable collections $\mathcal{U} \subseteq \mathbf{Meas}(\mathbb{R}, 2)$ are known as Borel-on-Borel [Kechris'87]
- Using descriptive set theory: If $\mathcal{U} \subseteq \mathbf{Meas}(\mathbb{R}, 2)$ is Borel-on-Borel and $\emptyset \in \mathcal{U}$
  then $\{x : \{x\} \notin \mathcal{U}\}$ is at most countable.

Main result

- Stark's ν-calculus [Stark'93] = higher-order programming with names
- We can **translate ν-calculus into a higher-order PPL** with continuous distributions where
  – names become real numbers
  – fresh name generation becomes sampling
- Privacy can be very subtle, e.g. compare

```
let a = normal(0,1) in
let b = normal(0,1) in
  fun x -> if x == b then b else a
```

– a is revealed

– b remains private

vs

```
let a = normal(0,1) in
let b = normal(0,1) in
  fun x -> if x == a then b else a
```

– a is revealed

– a can then be used to reveal b

– for this, the function must be called twice

Theorem:

Quasi-Borel space semantics is fully abstract for ν-calculus up to first-order function types

- **Full abstraction:** Two name generating programs are equivalent if and only if the corresponding probabilistic programs are equivalent, e.g. `(A*)` and `(A)`

Conclusions

- tl;dr `gensym = rnd`
- Name generation and probabilistic programming have interesting connections
- Random samples are a good semantics for fresh names
  – Was folklore for ground programs (various gensym implementations, randomized GUIDs)
  – We proved that the semantics is abstract even when higher-order functions are involved
  – Unified semantics of probability + names, more refined than traditional semantics (Nominal sets)
- Name-generation ideas like privacy naturally appear when reasoning about probabilistic programs
  – Randomization = anonymization
  – Bayesian inference on function types must be limited, as not to leak private information