Modular Bayesian inference

Yufei Cai, Zoubin Ghahramani, Chris Heunen, Ohad Kammar,
Sean K. Moss, Klaus Ostermann, Adam Ścibior, Sam Staton,
Matthijs Vákár, and Hongseok Yang

Dagstuhl Seminar:
Algebraic effects go mainstream
24 April 2018

What is statistical probabilistic programming?

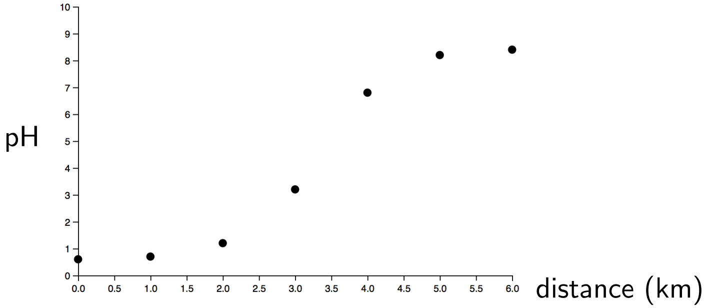
Bayesian data modelling

1. Develop a probabilistic (generative) model.
2. Design an inference algorithm for the model.
3. Using the algorithm, fit the model to the data.

What is statistical probabilistic programming?

Example

Acidity in soil



What is statistical probabilistic programming?

Generative model

$$
\begin{aligned}
s &\sim \mathsf{normal}(0, 2) \\
b &\sim \mathsf{normal}(0, 6) \\
f(x) &= s \cdot x + b \\
y_i &= \mathsf{normal}(f(i), 0.5) \\
&\qquad \text{for } i = 0 \ldots 6
\end{aligned}
$$

What is statistical probabilistic programming?

Generative model

$$\begin{aligned} s &\sim \mathsf{normal}(0, 2) \\ b &\sim \mathsf{normal}(0, 6) \\ f(x) &= s \cdot x + b \\ y_i &= \mathsf{normal}(f(i), 0.5) \\ &\qquad \text{for } i = 0 \ldots 6 \end{aligned}$$

Conditioning

$y_0 = 0.6, y_1 = 0.7, y_2 = 1.2, y_3 = 3.2, y_4 = 6.8, y_5 = 8.2, y_6 = 8.4$

Predict f?

What is statistical probabilistic programming?

Bayesian inference

"Bayes Law: $P(s, b|y_0, \ldots, y_6) = \dfrac{P(y_0, \ldots, y_6|s, b) \cdot P(s, b)}{P(y_0, \ldots, y_6)}$ "

What is statistical probabilistic programming?

Bayesian inference

Bayesian statistics:

$$\text{“ } posterior(s, b) \propto likelihood(y_0, \ldots, y_6 | s, b) \cdot prior(s, b) \text{ ”}$$

What is statistical probabilistic programming?

Bayesian inference

Bayesian statistics:

$$\text{“} \; posterior(s,b) \propto likelihood(y_0,\ldots,y_6|s,b) \cdot prior(s,b) \; \text{”}$$

$$posterior(s \leq s_0, b \leq b_0) \propto$$

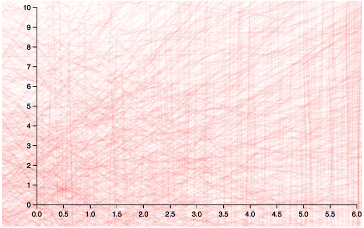
$$\int_{-\infty}^{s_0} \mathrm{d}s e^{-\frac{s^2}{2\cdot 2^2}} \int_{-\infty}^{b_0} \mathrm{d}b e^{-\frac{b^2}{2\cdot 6^2}} \prod_{i=0}^{6} e^{-\frac{(sx_i+b-y_i)^2}{2\cdot \frac{1}{2}^2}}$$

What is statistical probabilistic programming?

Bayesian inference

Bayesian statistics:

$$\text{“}\ posterior(s, b) \propto likelihood(y_0, \ldots, y_6 | s, b) \cdot prior(s, b)\ \text{”}$$



for $i = 1..1000$:

$(s, b) \sim prior$

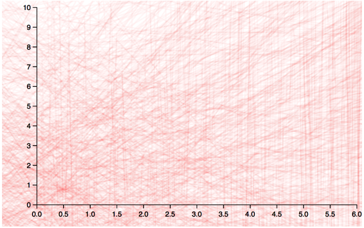
plot $sx + b$

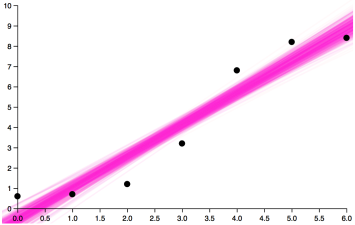
What is statistical probabilistic programming?

Bayesian inference

Bayesian statistics:

$$\text{“ } posterior(s,b) \propto likelihood(y_0,\ldots,y_6|s,b) \cdot prior(s,b) \text{ ”}$$





for $i = 1..1000$:

$(s,b) \sim prior$ $\qquad$ $(s,b) \sim posterior$

plot $sx + b$

What is statistical probabilistic programming?

Statistical probabilistic programming

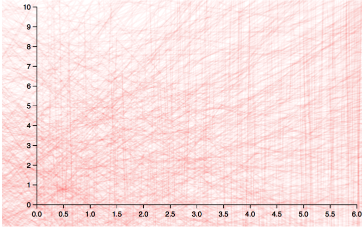
1. Generative models as probabilistic programs simultaneously manipulating the:
   (a) prior; and (b) liklihood
   (the fundamental concepts in Bayesian statistics)
2. ~~Design an inference algorithm for the model.~~
3. Using built-in algorithms, approximate the posterior.

What is probabilistic programming?

In Anglican [Wood et al.'14]

```
(let [s (sample (normal 0.0 2.0))
      b (sample (normal 0.0 6.0))
      f (fn [x] (+ (* s x) b))]
```

```
(predict :f f))
```



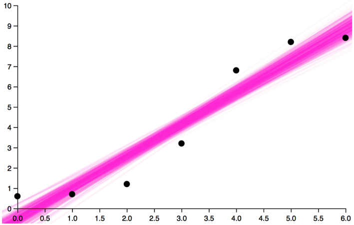
What is probabilistic programming?

In Anglican [Wood et al.'14]

```
(let [s (sample (normal 0.0 2.0))
      b (sample (normal 0.0 6.0))
      f (fn [x] (+ (* s x) b))]

  (observe (normal (f 1.0) 0.5) 2.5)
  (observe (normal (f 2.0) 0.5) 3.8)
  (observe (normal (f 3.0) 0.5) 4.5)
  (observe (normal (f 4.0) 0.5) 6.2)
  (observe (normal (f 5.0) 0.5) 8.0)

  (predict :f f))
```

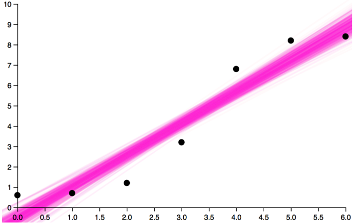


What is probabilistic programming?

In Anglican [Wood et al.'14]

```
(let [sample_linear
        (fn [] (let [s (sample (normal 0.0 2.0))
                     b (sample (normal 0.0 6.0))]
                 (fn [x] (+ (* s x) b))))
      f (sample_linear)]
  (observe (normal (f 1.0) 0.5) 2.5)
  (observe (normal (f 2.0) 0.5) 3.8)
  (observe (normal (f 3.0) 0.5) 4.5)
  (observe (normal (f 4.0) 0.5) 6.2)
  (observe (normal (f 5.0) 0.5) 8.0)

  (predict :f f))
```

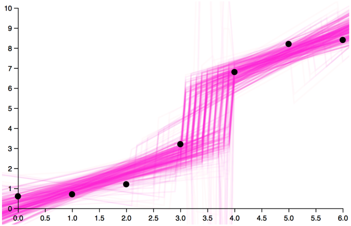


What is probabilistic programming?

In Anglican [Wood et al.'14]

```
(let [sample_linear
        (fn [] (let [s (sample (normal 0.0 2.0))
                     b (sample (normal 0.0 6.0))]
          (fn [x] (+ (* s x) b))))
      f (add-change-points sample_linear 0 6)]
  (observe (normal (f 1.0) 0.5) 2.5)
  (observe (normal (f 2.0) 0.5) 3.8)
  (observe (normal (f 3.0) 0.5) 4.5)
  (observe (normal (f 4.0) 0.5) 6.2)
  (observe (normal (f 5.0) 0.5) 8.0)

  (predict :f f))
```

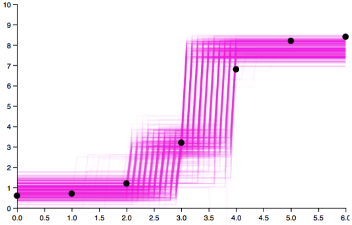


What is probabilistic programming?

In Anglican [Wood et al.'14]

```
(let [sample_linear
        (fn [] (let [
                       b (sample (normal 0.0 6.0))]
          (fn [x]      b )))
      f (add-change-points sample_linear 0 6)]
  (observe (normal (f 1.0) 0.5) 2.5)
  (observe (normal (f 2.0) 0.5) 3.8)
  (observe (normal (f 3.0) 0.5) 4.5)
  (observe (normal (f 4.0) 0.5) 6.2)
  (observe (normal (f 5.0) 0.5) 8.0)

  (predict :f f))
```



What is probabilistic programming?

High-level analogy

$$\overbrace{\text{graph algorithms}}^{\text{graph theorist}} + \overbrace{\text{graph library}}^{\substack{\text{graph savvy} \\ \text{hacker}}} + \overbrace{\text{graph manipulating program}}^{\text{user}}$$

$$\underbrace{\text{inference algorithms}}_{\text{statistician}} + \underbrace{\text{inference library}}_{\substack{\text{statistics savvy} \\ \text{hacker}}} + \underbrace{\text{probabilistic program}}_{\text{user}}$$

What is probabilistic programming?

Two effects

- Continuous probabilistic choice over the unit interval $\mathbb{I} := [0, 1]$:

$$\text{sample} : \mathbb{I}$$

- Conditioning:

$$\text{score} : \mathbb{R}_+ \to 1 \qquad \text{observe}(\text{normal}(a, b), x) := \frac{1}{\sqrt{2\pi b^2}} e^{-\frac{(x-a)^2}{2b^2}}$$

- Monadic semantics: MX is (s-finite) distributions over X:

$$\text{return}\, x_0 := \delta_{x_0} \qquad \forall a : X \to \mathbb{R}_+. \int_{\mathbb{R}_+} a(x)\delta_{x_0}(\mathrm{d}x) = a(x_0)$$

$$\mu \gg= f := \nu \qquad \forall a. \int a(x)\nu(\mathrm{d}y) = \int \mu(\mathrm{d}x) \int a(y) f(x)(\mathrm{d}y)$$

$$\text{sample} := \mathbf{U}_{\mathbb{I}} \qquad \text{score}\, r := r \cdot \delta_\star$$

Why is it hard?

Computing distributions

For $t : X$ we want to:

- Plot $[\![t]\!]$.
- Sample $[\![t]\!]$ (e.g., to make prediction)

Challenge 1: Integrals are hard to compute!

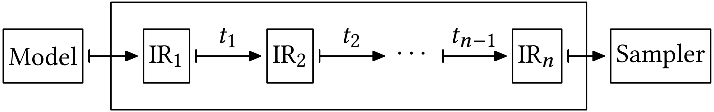
This talk: approximate using probabilistic simulation (Monte Carlo methods)
Complementary: use symbolic solvers (Maple, MatLab) as in Hakaru [Narayanan, Carette, Romano, Shan, and Zinkov, 2016]

Challenge 2

Given a fair coin ($\frac{1}{2}\delta_1 + \frac{1}{2}\delta_0$), how do we sample from a biased coin ($p\delta_1 + (1-p)\delta_0$)?
Generalise:
Given a prior distribution prior $[\![t]\!]$, how do we sample from $[\![t]\!]$?

What is inference?

Inference engine



Correctness of inference

Inference algorithm: distribution/meaning preserving transformation from one inference representation to another

What is inference?

Challenge 3

- Represented data is continuous
- Compositional inference representations (IRs)
- IRs are **higher-order**

What is inference?

Challenge 3

- Represented data is continuous
- Compositional inference representations (IRs)
- IRs are **higher-order**

Traditional measure theory is unsuitable:

Theorem (Aumann'61)

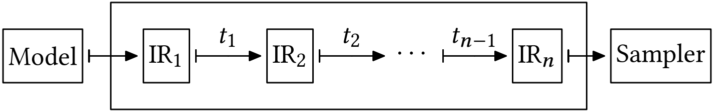
The set $\mathbf{Meas}(\mathbb{R}, \mathbb{R})$ *cannot be made into a measurable space with*

$$eval : \mathbf{Meas}(\mathbb{R}, \mathbb{R}) \times \mathbb{R} \to \mathbb{R}$$

measurable.

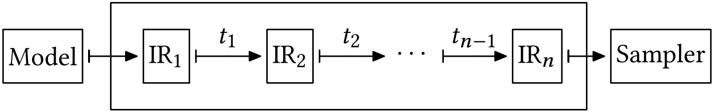
Contribution

Inference engine



Correctness of inference

- Modular validation of inference algorithms:
  Sequential Monte Carlo, Trace Markov Chain Monte Carlo
  By combining:
- Synthetic measure theory [Kock'12]: measure theory without measurable spaces
- Quasi-Borel spaces: a convenient category for higher-order measure theory [LICS'17]



Program representation

A **representation** $\underline{T} = (T, \mathrm{return}^{\underline{T}}, \gg\!=^{\underline{T}}, m^{\underline{T}})$ consists of:

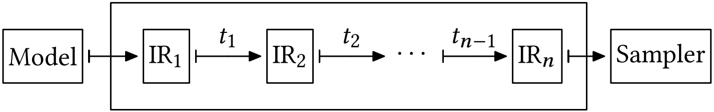
- $(T, \mathrm{return}^{\underline{T}}, \gg\!=^{\underline{T}})$: monadic interface;
- $m^{\underline{T}}_X : T\,X \to \mathrm{M}\,X$: meaning morphism for every space X

and $m^{\underline{T}}$ preserves $\mathrm{return}^{\underline{T}}$ and $\gg\!=^{\underline{T}}$:

$$m(\mathrm{return}^{\underline{T}}\,x) = \mathrm{return}^{\mathrm{M}}\,x = \delta_x$$

$$m(a \gg\!=^{\underline{T}} f) = (m\,a) \gg\!=^{\mathrm{M}} \lambda x.\ m(f\ x) = \int m(f\ x)\,m\,a(\mathrm{d}x)$$

Representations



Example representation: lists

$$
\begin{array}{ll}
\multicolumn{2}{l}{\textbf{instance}\ \mathit{Rep}\ (\mathsf{List})\ \textbf{where}} \\
\quad \textbf{return}\, x & = [x] \\
\quad x_s \gg\!= f & = \mathsf{foldr}\ [\ ] \\
 & \qquad (\lambda(x, y_s). \\
 & \qquad f(x) +\!\!+\ y_s)\ x_s \\
\quad m_{\mathsf{List}}[x_1, \ldots, x_n] & = \sum_{i=1}^{n} \underline{\delta}_{x_i}
\end{array}
$$

Example representation: lists

$$\begin{aligned}
&\textbf{instance}\ \mathit{Rep}\ (\mathsf{List})\ \textbf{where}\\
&\quad \textbf{return}\, x \qquad\qquad = [x]\\
&\quad x_s \gg\!= f \qquad\quad = \mathsf{foldr}\ [\ ]\\
&\qquad\qquad\qquad\qquad\qquad\quad (\lambda(x, y_s).\\
&\qquad\qquad\qquad\qquad\qquad\quad f(x) \mathbin{+\!\!+} y_s)\ x_s\\
&\quad m_{\mathsf{List}}[x_1, \ldots, x_n] = \textstyle\sum_{i=1}^{n} \underline{\delta}_{x_i}
\end{aligned}$$

$$m_{\mathsf{List}}[x] = \delta_x$$

Representations

Example representation: lists

$$\begin{aligned}
&\textbf{instance } \mathit{Rep}\ (\mathsf{List})\ \textbf{where}\\
&\quad \textbf{return}\, x = [x]\\
&\quad x_s \gg\!= f = \mathsf{foldr}\,[\,]\ (\lambda(x, y_s).\ f(x) +\!\!+\ y_s)\ x_s\\
&\quad m_{\mathsf{List}}[x_1, \ldots, x_n] = \textstyle\sum_{i=1}^{n} \underline{\delta}_{x_i}
\end{aligned}$$

$$m_{\mathsf{List}}\left([x_1, \ldots, x_n] \gg\!=^{\mathsf{List}} f\right) = m\left(f(x_1) +\!\!+ \cdots +\!\!+ f(x_n)\right)$$

$$= \sum_{i=1}^{n} m\, f(x_i) = \sum_{i=1}^{n} \oint m_{\mathsf{List}} \circ f(y) \delta_{x_i}(\mathrm{d}y) = \oint m \circ f(y) \sum_{i=1}^{n} \delta_{x_i}(\mathrm{d}y)$$

$$= \oint m \circ f(y)\, m[x_1, \ldots, x_n](\mathrm{d}y) = m[x_1, \ldots, x_n] \gg\!=^{\mathrm{M}} (m \circ f)$$

Representations

Sampling representation

$(T, \mathrm{return}^{\underline{T}}, \gg\!=^{\underline{T}}, m^{\underline{T}}, \mathbf{sample}^{\underline{T}})$

- $(T, \mathrm{return}^{\underline{T}}, \gg\!=^{\underline{T}}, m^{\underline{T}})$: program representation
- $\mathbf{sample}^{\underline{T}} : \mathbb{1} \to T\,\mathbb{I}$

and $m^{\underline{T}} \circ \mathbf{sample}^{\underline{T}} = \mathbf{U}_{\mathbb{I}}$

Conditioning representation

$(T, \mathrm{return}^{\underline{T}}, \gg\!=^{\underline{T}}, m^{\underline{T}}, \mathrm{score}^{\underline{T}})$

- $(T, \mathrm{return}^{\underline{T}}, \gg\!=^{\underline{T}}, m^{\underline{T}})$: program representation
- $\mathrm{score}^{\underline{T}} : [0, \infty) \to T\,\mathbb{1}$

and $m^{\underline{T}} \circ \mathrm{score}^{\underline{T}}\, r = r \cdot \underline{\delta}_{()}$

Representations

Example: free sampler

$\mathsf{Sam}\,\alpha \coloneqq \{\mathsf{Return}\,\alpha \mid \mathsf{Sample}\,(\mathbb{I} \to \mathsf{Sam}\,\alpha)\}$:

$$
\begin{array}{l}
\textbf{instance}\ \textit{Sampling Rep}\ (\mathsf{Sam})\ \textbf{where} \\
\quad \textbf{return}\, x \;=\; \mathsf{Return}\, x \\
\quad a \gg\!= f \;=\; \textbf{match}\, a\, \textbf{with}\, \{ \\
\qquad\qquad \mathsf{Return}\, x \to f(x) \\
\qquad\qquad \mathsf{Sample}\, k \to \\
\qquad\qquad\quad \mathsf{Sample}\,(\lambda r.\, k(r) \gg\!= f)\} \\
\quad \mathrm{sample} \;=\; \mathsf{Sample}\,\lambda r.\,(\mathsf{Return}\, r) \\
\quad m\, a \;=\; \textbf{match}\, a\, \textbf{with}\, \{ \\
\qquad\qquad \mathsf{Return}\, x \to \underline{\delta}_x \\
\qquad\qquad \mathsf{Sample}\, k \to \oint_{\mathbb{I}} m(k(x))\mathbf{U}(\mathrm{d}x)\}
\end{array}
$$

Representations

Inference representation

$(T, \mathrm{return}^{\underline{T}}, \gg\!=^{\underline{T}}, \mathbf{sample}^{\underline{T}}\, \mathrm{score}^{\underline{T}}, m^{\underline{T}})$: sampling and conditioning

Example: weighted sampler

$\mathsf{WSam}\, X := \mathsf{W}\, \mathsf{Sam}\, X = \mathsf{Sam}([0, \infty) \times X)$

Inference transformations

$\underline{t} : \underline{T} \to \underline{S}$

$\underline{t} : T\,X \to S\,X$ for every space X such that:

$$m_{\underline{S}} \circ \underline{t} = m_{\underline{T}}$$

A single compositional step in an inference algorithm

Inference transformations

$\underline{t} : \underline{T} \to \underline{S}$

$\underline{t} : T\,X \to S\,X$ for every space X such that:

$$m_{\underline{S}} \circ \underline{t} = m_{\underline{T}}$$

A single compositional step in an inference algorithm

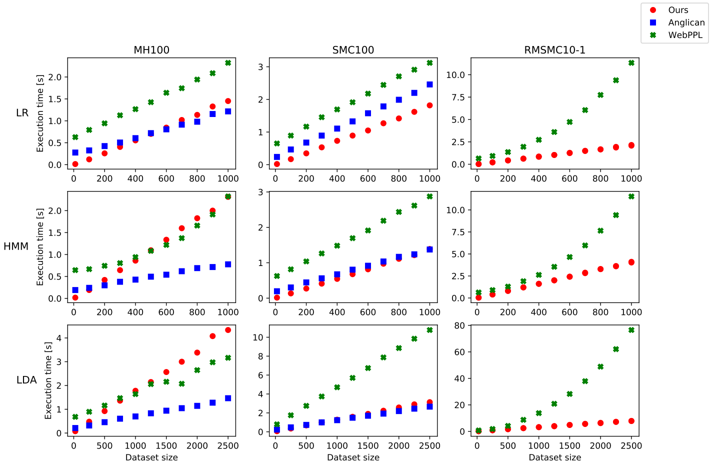
Unnaturality

$\mathrm{aggr}_X : \mathsf{List}(\mathbb{R}_+ * X) \to \mathsf{List}(\mathbb{R}_+ * X)$
aggregating (r, x), (s, x) to $(r + s, x)$
Then $\mathrm{aggr} : \underline{\mathsf{List}} \to \underline{\mathsf{List}}$ but not natural:

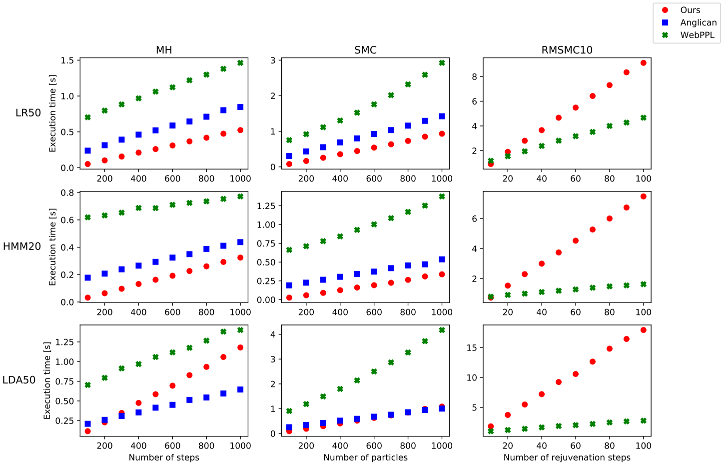
$$\begin{aligned} \mathrm{aggr} \circ \mathsf{List}!\ [(\tfrac{1}{2}, \mathsf{False}), (\tfrac{1}{2}, \mathsf{True})] &= \mathrm{aggr}\ [(\tfrac{1}{2}, ()), (\tfrac{1}{2}, ())] \\ &= [(1, ())] \neq [(\tfrac{1}{2}, ()), (\tfrac{1}{2}, ())] \\ = \mathsf{Enum}!\ [(\tfrac{1}{2}, \mathsf{False}), (\tfrac{1}{2}, \mathsf{True})] &= \mathsf{Enum}! \circ \mathrm{aggr}\ [(\tfrac{1}{2}, \mathsf{False}), (\tfrac{1}{2}, \mathsf{True})] \end{aligned}$$

MonadBayes: Modular implementation in Haskell

Performance evaluation (1)

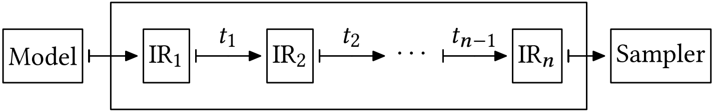


Performance evaluation (2)



Contribution

Inference engine



Correctness of inference

- Modular validation of inference algorithms:
  Sequential Monte Carlo, Trace Markov Chain Monte Carlo
  By combining:
- Synthetic measure theory [Kock'12]: measure theory without measurable spaces
- Quasi-Borel spaces: a convenient category for higher-order measure theory [LICS'17]

Conclusion

Summary

- Bayesian inference: (continuous) sampling and conditioning
- Inference representation: monadic interface, sampling, conditioning, and meaning
- Plenty of opportunities for traditional programming language expertise

Further topics

- Sequential Monte Carlo (SMC)
- Markov Chain Monte Carlo (MCMC) and Metropolis-Hastings-Green Theorem for $\mathbf{Qbs}$
- Combining SMC and MCMC into Move-Resample SMC