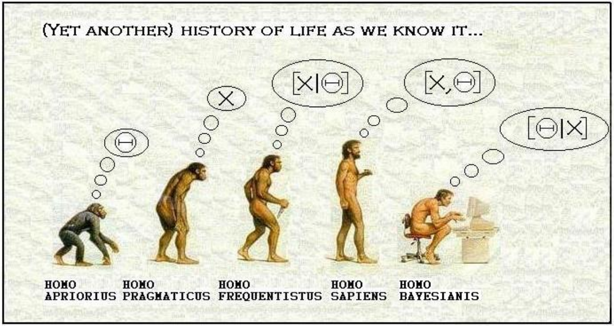
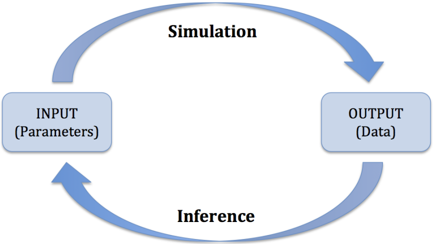
Probabilistic Programming Languages: *Bayesian Inference*

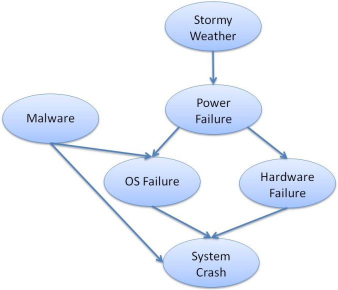
Holakou Rahmanian & Parameswaran Raman

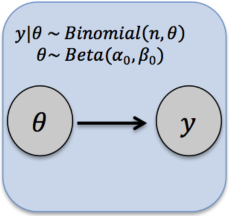
University of California, Santa Cruz



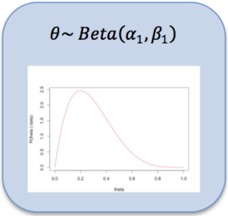
Bayesian Inference and PPLs







#	y
1	12
2	10
3	8
4	17
5	40
6	8
7	2



Model

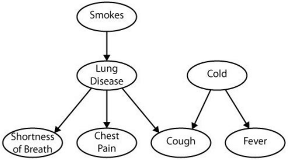
Data

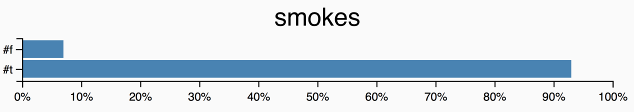
Parameters

PPL Example - Church

- Developed at MIT in 2008 - named for computation pioneer *Alonzo Church*
- Universal language for describing stochastic generative processes
- Based on the Lisp model of λ-calculus
- Different implementations: *Webchurch, Bher Church, MIT-Church, Cosh*

```
(define samples
  (mh-query 200 100
    (define smokes (flip 0.2))

    (define lung-disease (or (flip 0.001) (and smokes (flip 0.1))))
    (define cold (flip 0.02))

    (define cough (or (and cold (flip 0.5)) (and lung-disease (flip 0.5)) (flip 0.001)))
    (define fever (or (and cold (flip 0.3)) (flip 0.01)))
    (define chest-pain (or (and lung-disease (flip 0.2)) (flip 0.01)))
    (define shortness-of-breath (or (and lung-disease (flip 0.2)) (flip 0.01)))

     smokes

     (and cough chest-pain shortness-of-breath)
  )
)
(hist samples "smokes")
```





PPL Example - FACTORIE

- FACTORIE a.k.a "Factor graphs, Imperative, Extensible", Developed at University of Massachusetts Amherst in 2009
- Written in and uses Scala as the programming language.
- Creating Factor graphs, estimating parameters and performing inference

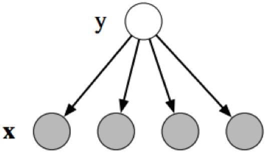
```
object TutorialSimpleChain extends App {
  // Imports Inference Methods, other required types here ...
  import cc.factorie.infer.{ GibbsSampler, InferByBPChain }

  implicit val random = new scala.util.Random(0)

  object LabelDomain extends CategoricalDomain[String]
  class Label(val token: Token, s: String) extends LabeledCategoricalVariable(s) {
    ...
  }
  object FeaturesDomain extends CategoricalVectorDomain[String]
  class Features(val token: Token) extends BinaryFeatureVectorVariable[String] {
    ...
  }
```

Setup the model

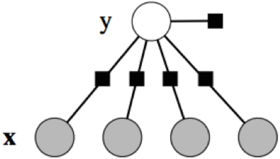
```
  object model extends ChainModel[Label, Features, Token](
    ...

  // The Document class implements documents as sequences of sentences and tokens.
  val document = new Document("The quick brown fox jumped over the lazy dog.")
  val tokenizer = new app.nlp.segment.DeterministicTokenizer
  tokenizer.process(document)
  val segmenter = new app.nlp.segment.DeterministicSentenceSegmenter
  segmenter.process(document)
  assertStringEquals(document.tokenCount, "10")
  assertStringEquals(document.sentenceCount, "1")

  // Label the tokens and initialize features
  document.tokens.foreach(t => t.attr += new Label(t, "A"))
  LabelDomain.index("B")
  document.tokens.foreach(t => {
    val features = t.attr += new Features(t)
    features += "W=" + t.string.toLowerCase
    features += "IsCapitalized=" + t.string(0).isUpper.toString
  })
```

Inference

```
  val summary = InferByBPChain.infer(document.tokens.toSeq.map(_.attr[Label]), model)
  assertStringEquals(summary.logZ, "6.931471805599453")
  assertStringEquals(summary.marginal(document.tokens.head.attr[Label]).proportions,
                     "Proportions(0.49999999999999994,0.49999999999999994)")
```



Graphical Model Representation



Factor Graph Representation

PPLs Comparison

BUGS

- Gibbs Sampling, Propositional Logic
- Pros: Simple
- Cons: Not scalable

Church

- Metropolis-Hasting, Lisp, λ-calculus, generative
- Pros: Higher-order logic, Representational flexibility
- Cons: Inference complexity, inefficient implementations

Infer.NET

- Gibbs, EP, Variational Message Passing
- Pros: OOP, Scalable, Great Documentation
- Cons: Unrolling slows down inference

FACTORIE

- Imperative, Discriminative Models, full support for NLP pipeline
- Pros: OOP, Scalable, Parallelizable
- Cons: No support for Contin Random Vars, Insufficient Documentation