STEVENS
INSTITUTE of TECHNOLOGY
THE INNOVATION UNIVERSITY®

Process Generators

SYS 611: Systems Modeling and Simulation

Paul T. Grogan, Ph.D.
Assistant Professor
School of Systems and Enterprises

Agenda

1. Random Numbers
2. Discrete Process Generators
3. Continuous Process Generators

Reference: S.M. Ross, “Elements of Probability,” “Random Numbers,” Ch. 2-3 in *Simulation*, 2012.

J.V. Farr, “Review of Probability and Statistics,” Ch. 3 in Simulation of Complex Systems and Enterprises, Stevens Institute of Technology, 2007.

Random Numbers

Random Number Generators

- **Pseudorandom numbers** can be generated using a computational algorithm (generator)
  - Deterministic sequences of random variables
  - Often seeded with controlled initial conditions
  - Uniform U(0,1) is most common PDF provided
- Most software libraries today have good RNGs
- Hardware generators may use aleatory data sources
  - Thermal noise
  - Quantum phenomena

Sun Microsystems Crypto Accelerator
(Shieldforyoureyes/Wikimedia)

RNG in Modeling Tools

In Excel:

- Uniform (0,1) distribution: **`=RAND()`**
- As of Excel 2010, uses Mersenne Twister algorithm (MT19937)
- Regenerates all random variables on any change
- No random number seed

In Python:

- Default or Numpy library:
```
import random
random.rand()
import numpy as np
np.random.rand()
```
- Both use Mersenne Twister algorithm (MT19937)
- Random number seed: **`(np.)random.seed(0)`**
- Many more functions…

Discrete Process Generators

Discrete Process Generators

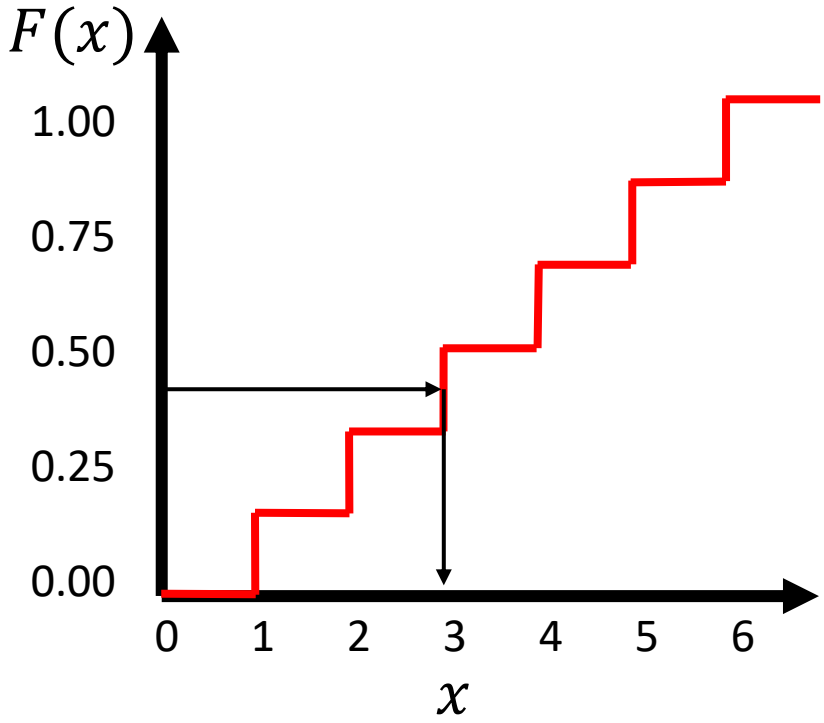
Discrete processes are a sequence of discrete random variable values

- Built-in process generators exist for some common distributions (Uniform, Binomial, etc.)
- Two methods to generate arbitrary processes:
  - **Inverse transform method (IVT)** – requires global knowledge of the CDF
  - Accept-reject method (ARM) – only requires local knowledge of the PMF but less efficient

IVT for Discrete Processes

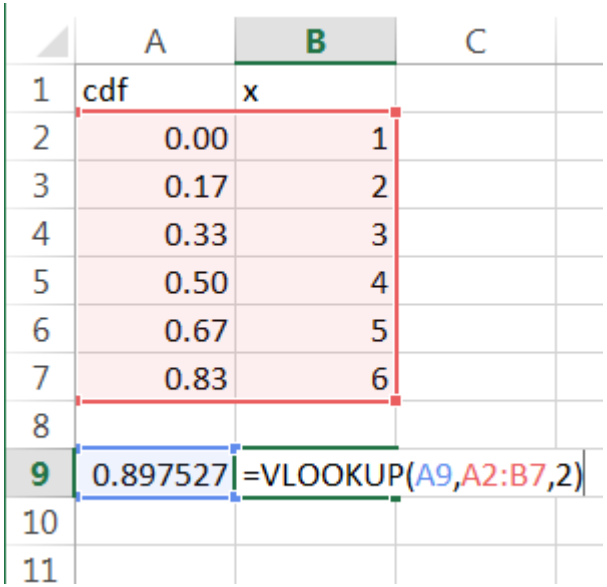
Generate U(0,1) value r

Draw horizontal line from r on y-axis to CDF and read x-value



Discrete IVT Generators in Excel

```
import numpy as np

def gen_roll_ivt():
    r = np.random.rand()
    if r < 1/6:
        roll = 1
    elif r < 2/6:
        roll = 2
    elif r < 3/6:
        roll = 3
    elif r < 4/6:
        roll = 4
    elif r < 5/6:
        roll = 5
    else:
        roll = 6
    return roll
```



- CDF lower bounds
- Random variable (x) values
- RNG (**=RAND()**)
- **VLOOKUP** function

Example: Café Java Demands

- Stevens students enjoy coffee at Café Java. The manager gathered data last week for coffee demand during the ~7:45pm break period.
- Discrete random variable X is the demand

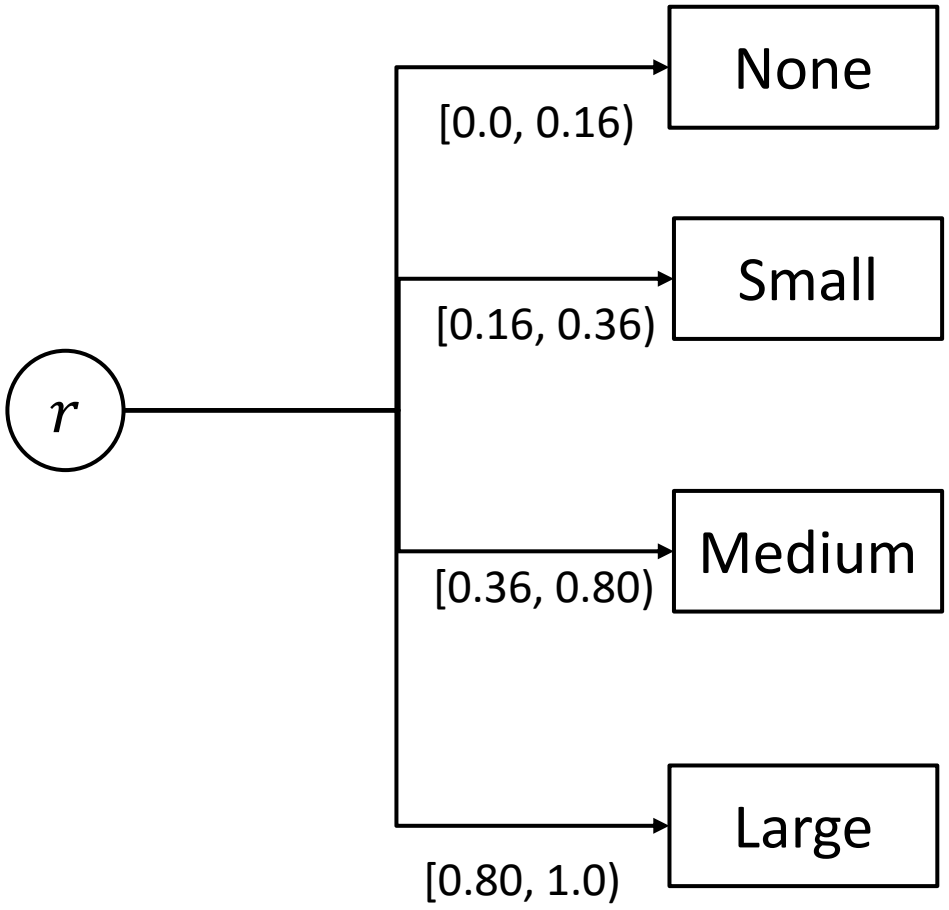
Demand (X)	Frequency
No coffee	8
Small	10
Medium	22
Large	10

- Find the PMF/CDF and develop a discrete process generator for demands in a future simulation.

Café Java Demands PMF/CDF

Demand (X)	Frequency	$P\{X = x\} = p(x)$	$P\{X \leq x\} = F(x)$
No coffee	8	=8/50 = 0.16	=8/50 = 0.16
Small	10	=10/50 = 0.20	=(8+10)/50 = 0.36
Medium	22	=22/50 = 0.44	=(8+10+22)/50 = 0.80
Large	10	=10/50 = 0.20	=(8+10+22+10)/50 =50/50 = 1.0

IVT for Café Example

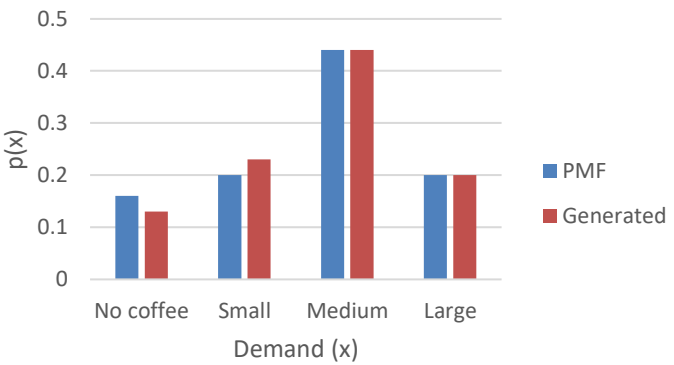


Demand IVT Generator in Excel

VLOOKUP Table

SUM | =VLOOKUP(B10,$G$3:$H$6,2)

	A	B	C	D	E	F	G	H
2	Demand	Frequency	PMF	CDF			CDF (LB)	Demand
3	No coffee	8	0.16	0.16			0	No coffee
4	Small	10	0.2	0.36			0.16	Small
5	Medium	22	0.44	0.8			0.36	Medium
6	Large	10	0.2	1			0.8	Large
7								
8	Sampled data							
9	i	r_i	x_i					
10	1	0.405569	=VLOOKUP(B10,$G$3:$H$6,2)					
11	2	0.485461	Medium					
12	3	0.577465	Medium					
13	4	0.984665	Large					

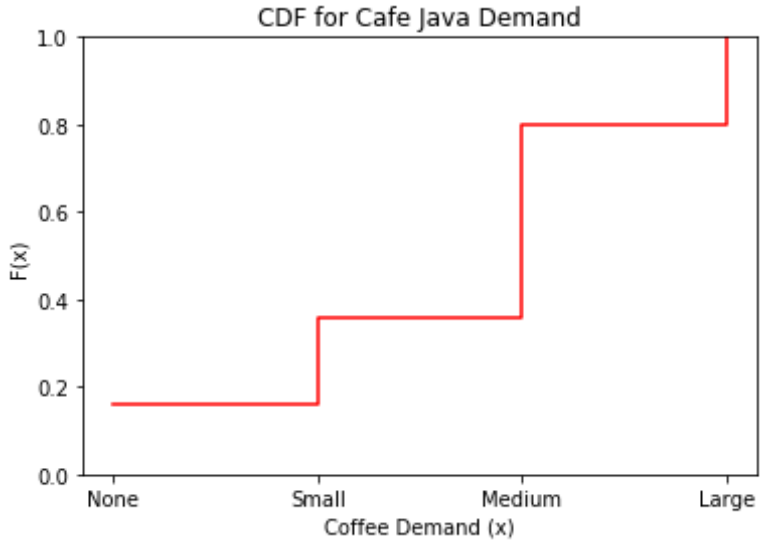


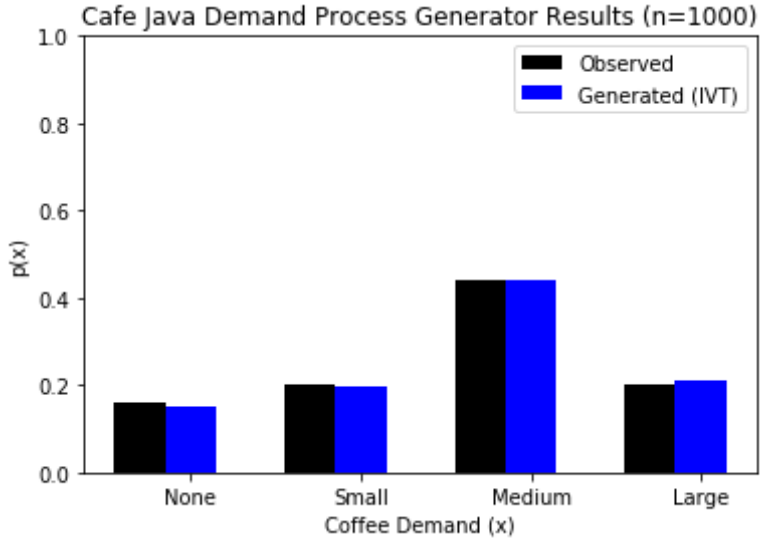
=RAND ( )

IVT Process Generator

Demand IVT Generator in Python

```
import numpy as np

demands = np.array([0, 1, 2, 3])
frequency = np.array([8, 10, 22, 10])

pmf = frequency/np.sum(frequency)
cdf = np.cumsum(pmf)

def gen_demand_ivt():
  r = np.random.rand()
  for i in range(len(demands)):
    if r <= cdf[i]:
      return i

samples_ivt = [gen_demand_ivt()
    for i in range(1000)]

counts = np.array(
  [sum(samples_ivt==i) for i in demands]
)
frequency_ivt = counts/np.sum(counts)
```





Continuous Process Generators

Continuous Process Generators

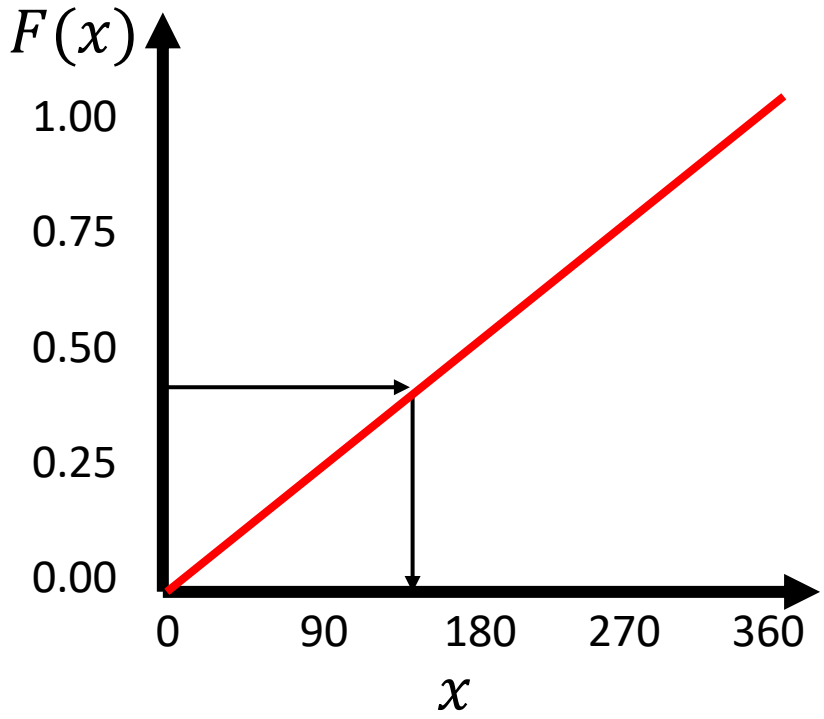
Continuous processes are a sequence of continuous random variable values

- Built-in process generators exist for most common distributions (Uniform, Normal, etc.)
- Two methods to generate arbitrary processes:
  - **Inverse transform method (IVT)** – requires global knowledge of the CDF
  - Accept-reject method (ARM) – only requires local knowledge of the PDF but less efficient

IVT for Continuous Processes

Generate U(0,1) value r

Equate r to CDF $F(x)$ and solve for x

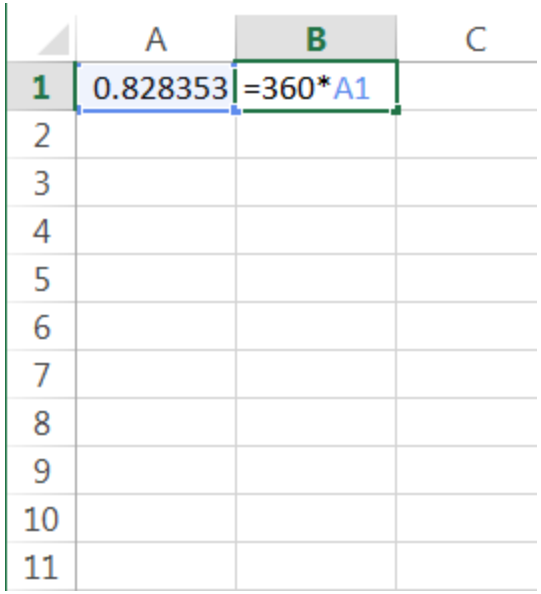


$$r = F(x) = \frac{x}{360}$$

$$\rightarrow x = 360 * r$$

Continuous IVT Generators

```
import numpy as np


def gen_spin_ivt():
    r = np.random.rand()
    return 360*r
```



- RNG (`=RAND()`)
- Inverse CDF

Example: Café Java Arrivals

The Café Java manager contacts Dr. Farr to model arrivals. Farr reports customers arrive as a Poisson process with a 2-minute inter-arrival period ($\lambda = 1/2$ customers per minute). Develop a continuous process generator for inter-arrival times.

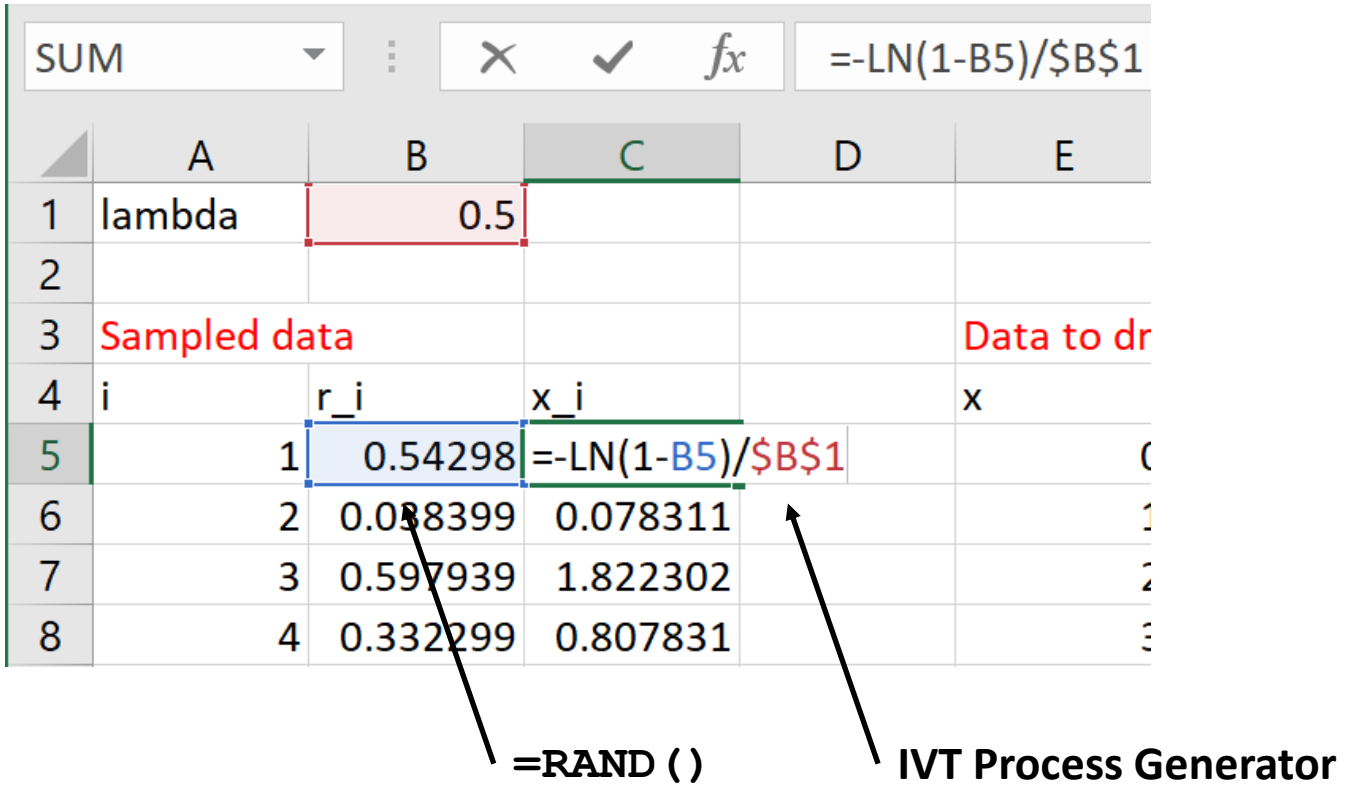
$$X\text{: time between customer arrivals}$$

$$X \sim \text{exponential}(\lambda = 1/2) \Rightarrow f(x) = \lambda e^{-\lambda x}$$

$$F(x) = \int_{i=0}^{x} f(i)di = \int_{i=0}^{x} \lambda e^{-\lambda i} di = 1 - e^{-\lambda x}$$

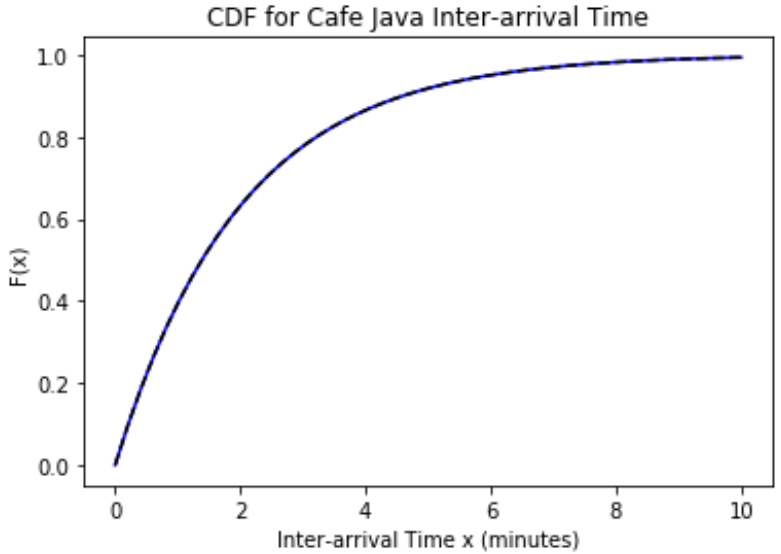
$$r = 1 - e^{-\lambda x} \Rightarrow -\lambda x = \ln(1-r) \Rightarrow x = \frac{-\ln(1-r)}{\lambda}$$

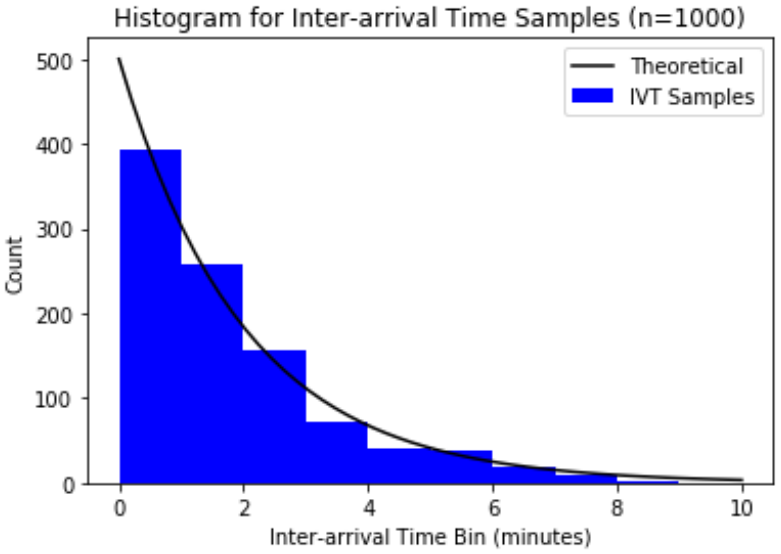
Arrivals IVT Generator in Excel



Arrivals IVT Generator in Python

```
import numpy as np

_lambda = 1/2

plot_x = np.linspace(0,10)
pdf = _lambda*np.exp(-_lambda*plot_x)
cdf = 1-np.exp(-_lambda*plot_x)

def gen_arrival_ivt():
  r = np.random.rand()
  return -np.log(1-r)/_lambda

num_samples = 1000
samples_ivt = [gen_arrival_ivt()
    for i in range(num_samples)]
```



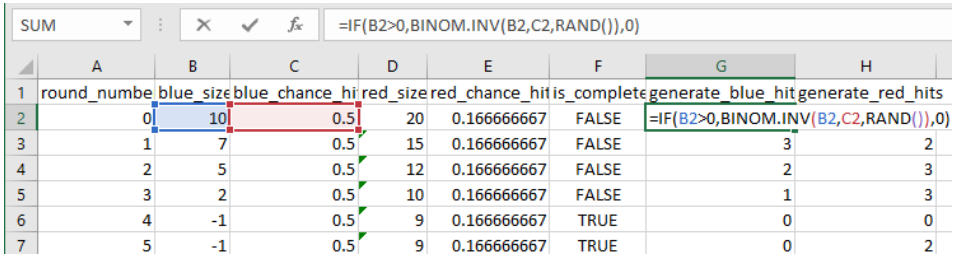


IVT/RNG in Modeling Tools

In Excel:

- Some inverse functions (INV) available:

```
=BINOM.INV(n, p, RAND())
=NORM.INV(RAND(), m, s)
```



	A	B	C	D	E	F	G	H
1	round_numbe	blue_size	blue_chance_hi	red_size	red_chance_hit	is_complete	generate_blue_hit	generate_red_hits
2	0	10	0.5	20	0.166666667	FALSE	=IF(B2>0,BINOM.INV(B2,C2,RAND()),0)	
3	1	7	0.5	15	0.166666667	FALSE	3	2
4	2	5	0.5	12	0.166666667	FALSE	2	3
5	3	2	0.5	10	0.166666667	FALSE	1	3
6	4	-1	0.5	9	0.166666667	TRUE	0	0
7	5	-1	0.5	9	0.166666667	TRUE	0	2

```
def generate_blue_hits():
  return stats.binom.ppf(
    np.random.rand(),
    blue_size, blue_chance_hit)

def generate_blue_hits():
  return np.random.binomial(
    blue_size, blue_chance_hit)
```

In Python:

- Inverse (percent-point) functions (ppf) from **`scipy.stats`** library:

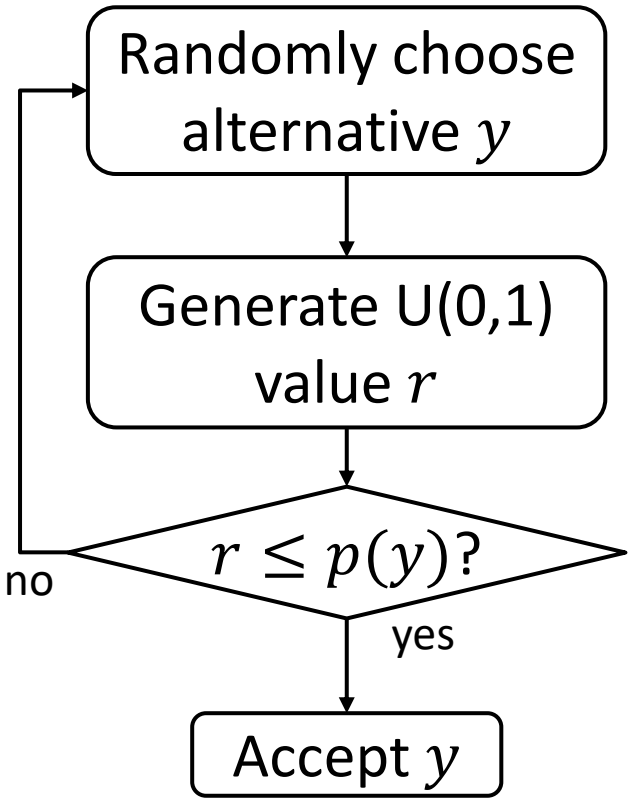
```
r=np.random.rand()
stats.binom.ppf(r,n,p)
stats.norm.ppf(r,m,s)
```

- Built-in generators:

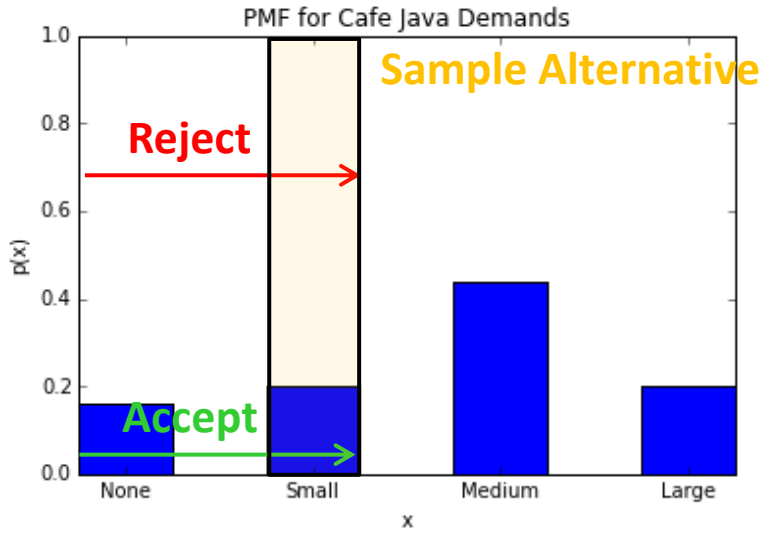
```
random.uniform(a,b)
random.triangular(a,b,c)
random.expovariate(l)
…
np.random.exponential(l)
np.random.binomial(n,p)
…
```

Accept-Reject Method

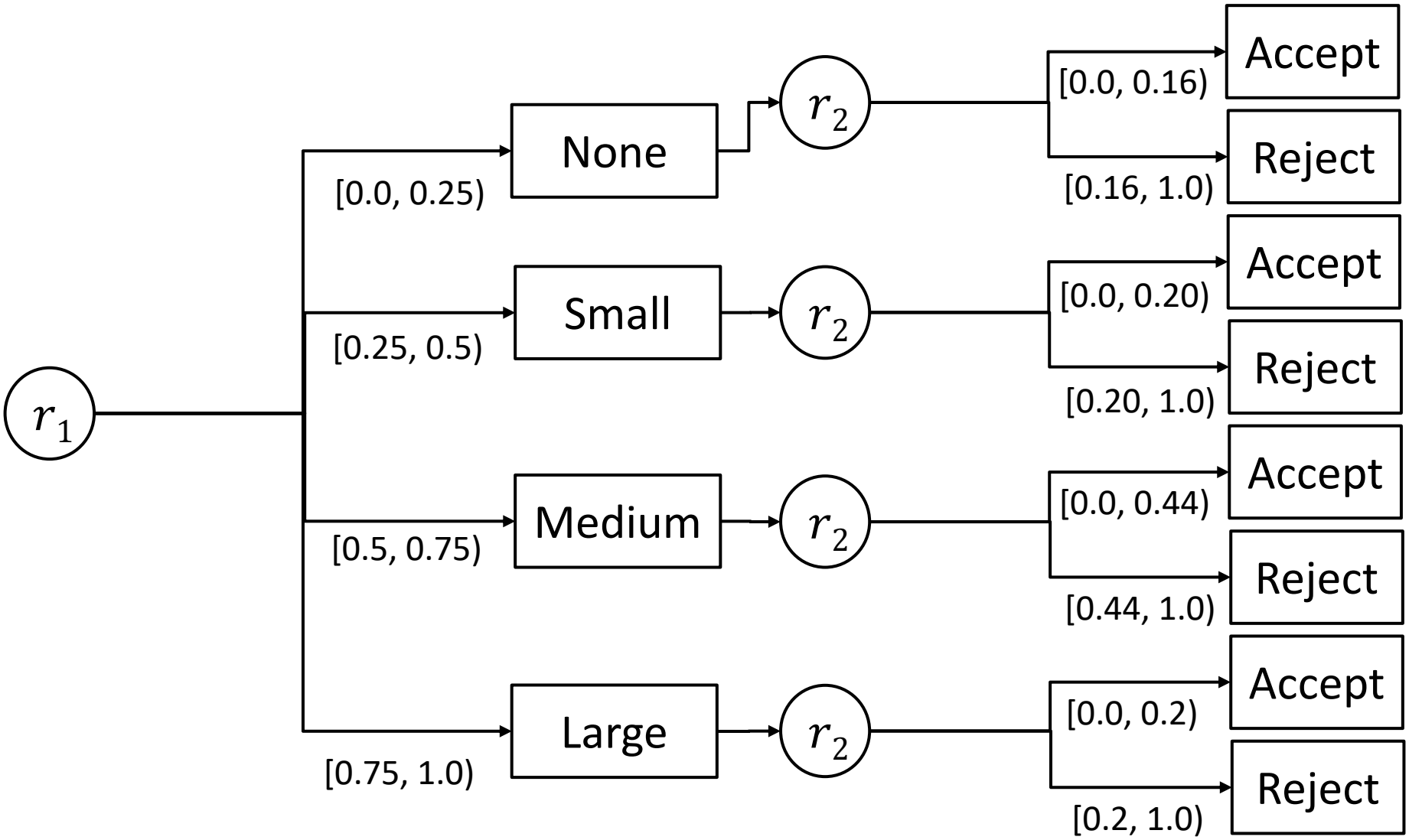
ARM for Discrete Processes



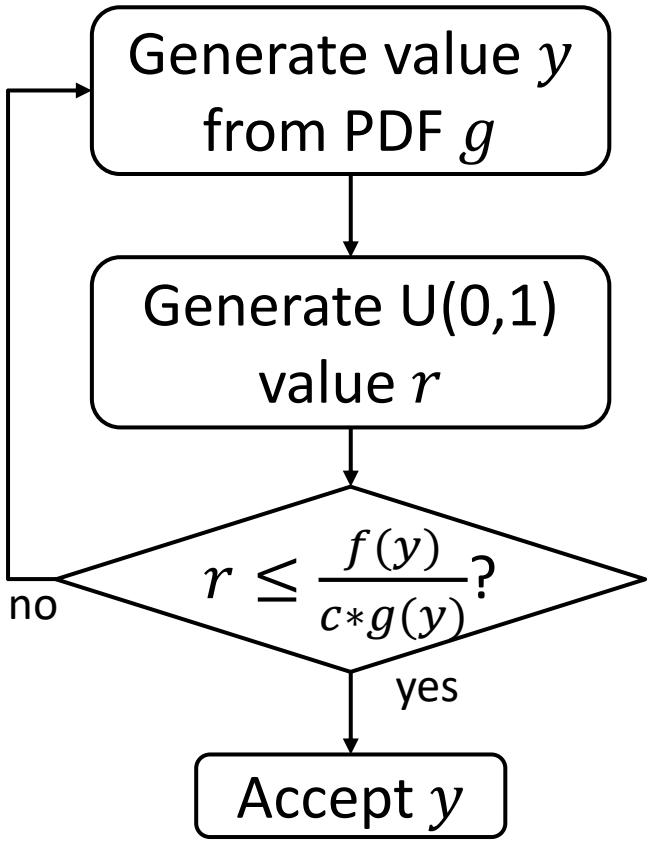
- Some CDFs are not easy to quantify or express
- Rely only on PMFs
- Example: Café Java



ARM for Café Demand Example



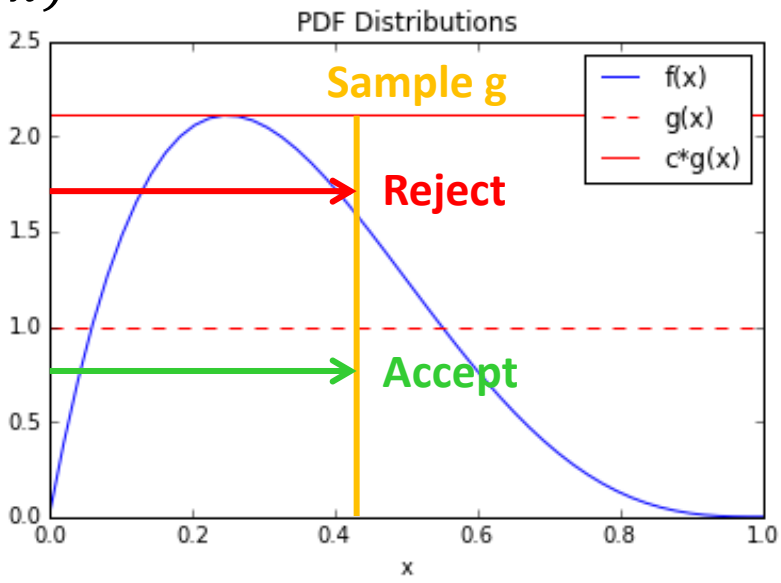
ARM for Continuous Processes



- Some CDFs do not have closed-form equations
- Rely only on PDFs
  - Use a simpler "enveloping" distribution $g(x)$ where $c * g(x) \geq f(x) \forall x$
  - Simplest: $g(x) \sim \text{uniform}(a, b)$
  - Find maximum $f(x)$ and assign c appropriately

Accept-Reject Example (Ross)

- PDF: $f(x) = 20x(1-x)^3, 0 < x < 1$
- Proposed PDF: $g(x) = 1, 0 < x < 1$
- What is the max value of $f(x)$ to ensure enveloping?
  - $0 = f'(x) = 20(1-x)^3 - 60x(1-x)^2$
  - $= -20(x-1)^2(4x-1)$
  - $\rightarrow f(0.25) = \frac{135}{64} \rightarrow c = \frac{135}{64}$
- $r \leq \frac{f(y)}{c \cdot g(y)} = \frac{256}{27} y(1-y)^3$
- Equivalently: $r \cdot c \cdot g(y) \leq f(y)$



ARM for Ross Example