Process Generators

SYS-611: Simulation and Modeling

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

Agenda

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

Reading: S.M. Ross, "Generating Discrete Random Variables," and "Generating Continuous Random Variables," Ch. 4-5 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
- Hardware generators may use aleatory data sources
  - Thermal noise
  - Quantum phenomena

Sun Microsystems Crypto Accelerator (Shieldforyoureyes/Wikimedia)

Copyright © 2018 Paul T. Grogan

Example: Human RNG

- Submit random numbers: `goo.gl/WaCZda`
  - What biases can be observed?
- Random number generators are critical to effective stochastic simulation
  - Eliminate any underlying biases
  - Reproducible/seeded streams help verification
- Most software libraries today have good RNGs

Copyright © 2018 Paul T. Grogan

Discrete Process Generators

Discrete Process Generators

Discrete processes are a sequence of discrete random variable values

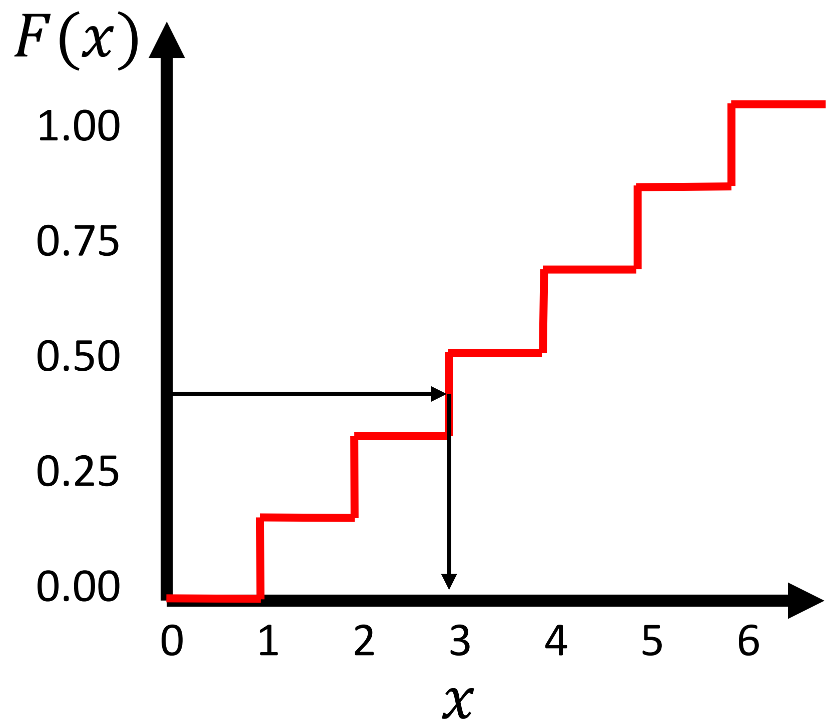
- Built-in process generators exist for most common distributions (Uniform, Binomial, etc.)
- Two methods to generate arbitrary processes:
  - Inverse transform method – requires global knowledge of the CDF
  - Accept-reject method – only requires local knowledge of the PMF but less efficient

Inverse Transform Method

Generate U(0,1) value r

↓

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



Copyright © 2018 Paul T. Grogan

Inverse Transform for Discrete Processes

```
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
```

	A	B	C
1	cdf	x	
2	0.00	1	
3	0.17	2	
4	0.33	3	
5	0.50	4	
6	0.67	5	
7	0.83	6	
8			
9	0.897527	=VLOOKUP(A9,A2:B7,2)	
10			
11			

- CDF lower bounds
- RV (x) values
- RNG (`=RAND()`)
- `VLOOKUP` function

Class Problem: Café Java

- Stevens students enjoy coffee at Café Java. The manager gathered data last week for coffee demand during the ~7:30pm break period.

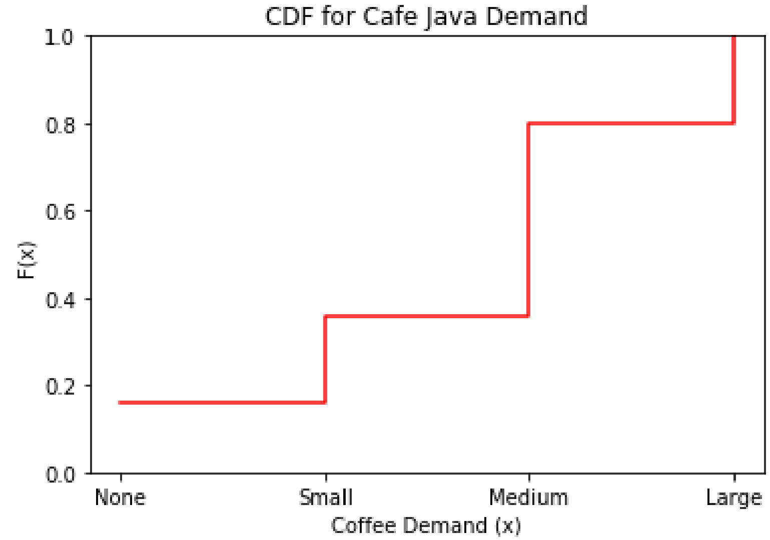
Demand (X)	Frequency	$P\{X = x\} = p(x)$	$P\{X \leq x\} = F(x)$
No coffee	8		
Small	10		
Medium	22		
Large	10		

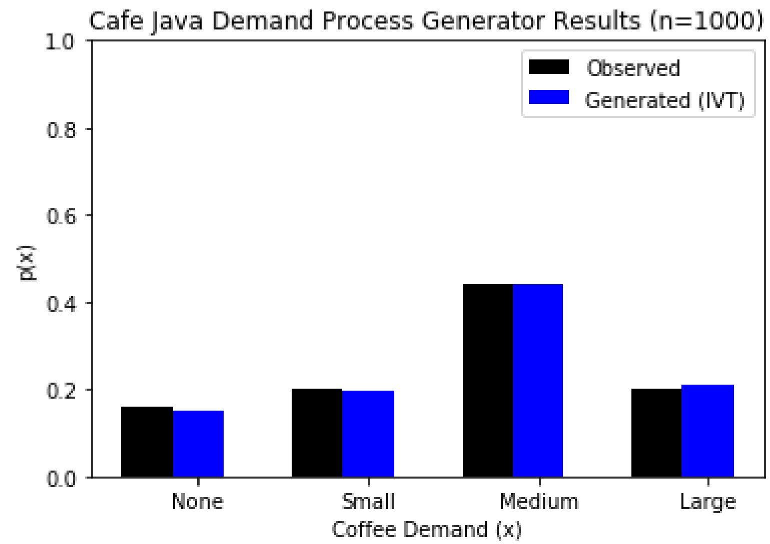
- Complete the PMF and CDF and develop a discrete process generator for future simulation.

Copyright © 2018 Paul T. Grogan

Café Java Demand Generators

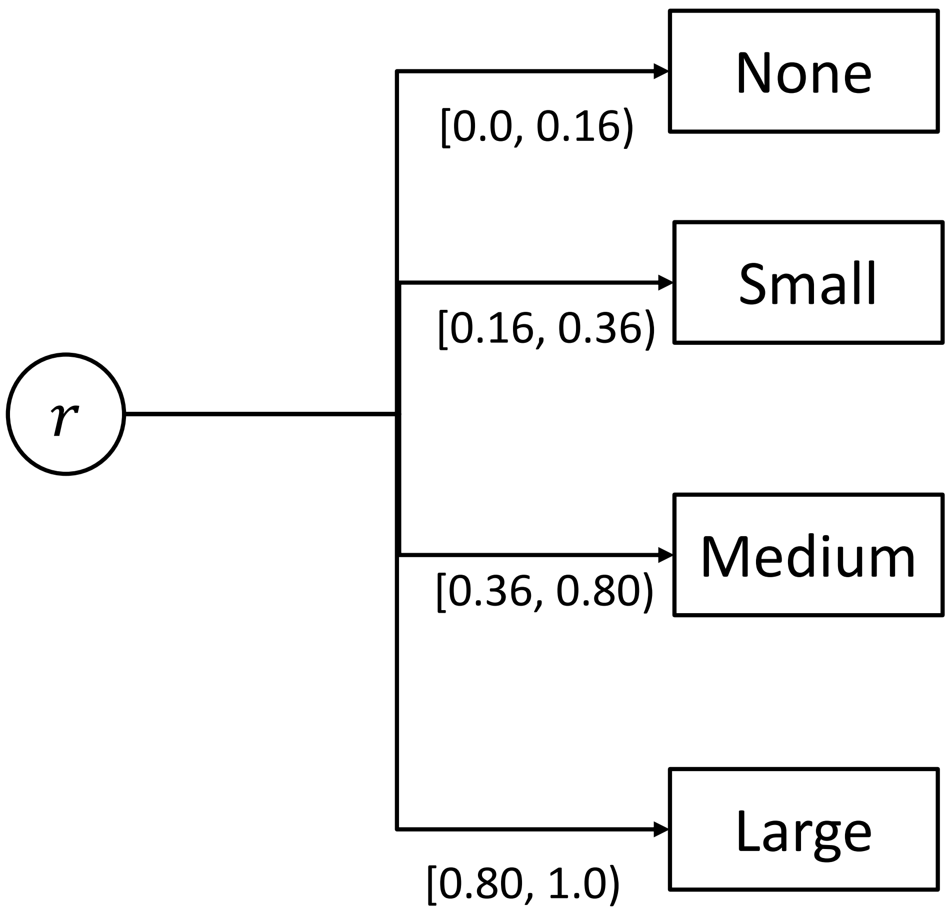
```
import numpy as np

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

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

def gen_demand_ivt():
  r = np.random.rand()
  for i in 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/float(np.sum(counts))
```





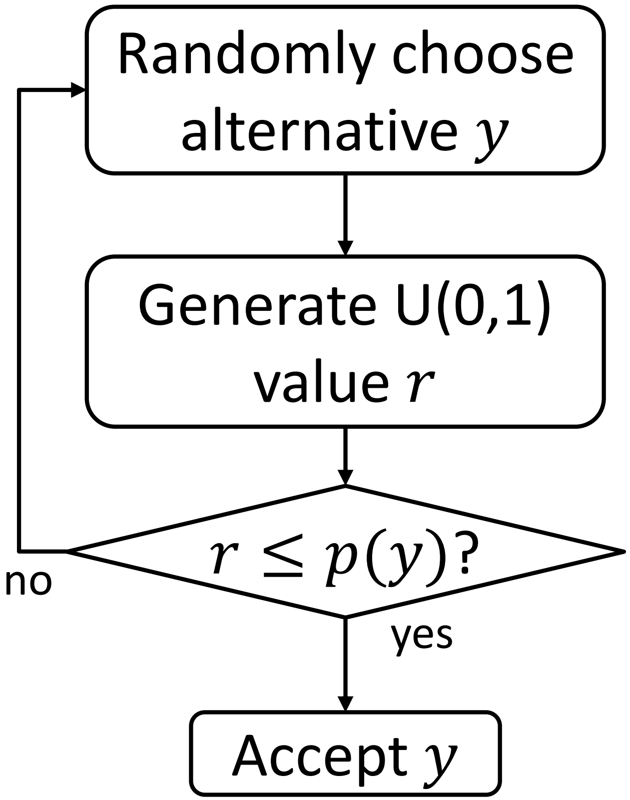
Copyright © 2018 Paul T. Grogan

Inverse Transform for Discrete Processes

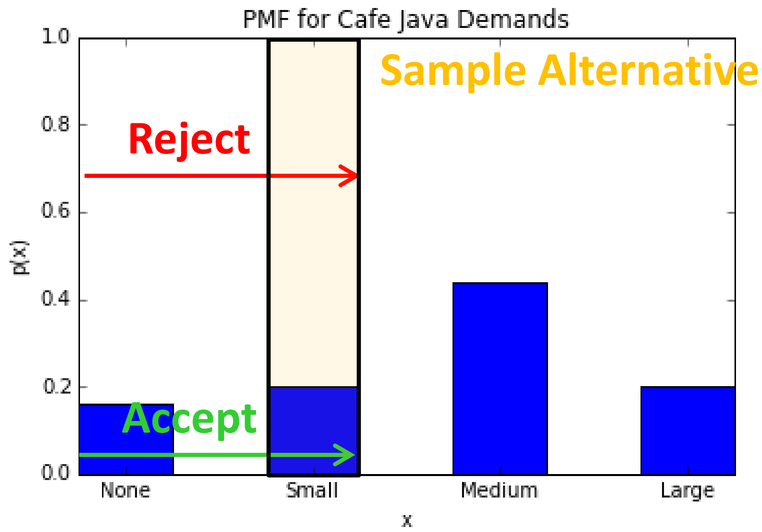


Copyright © 2018 Paul T. Grogan

Accept-Reject Method

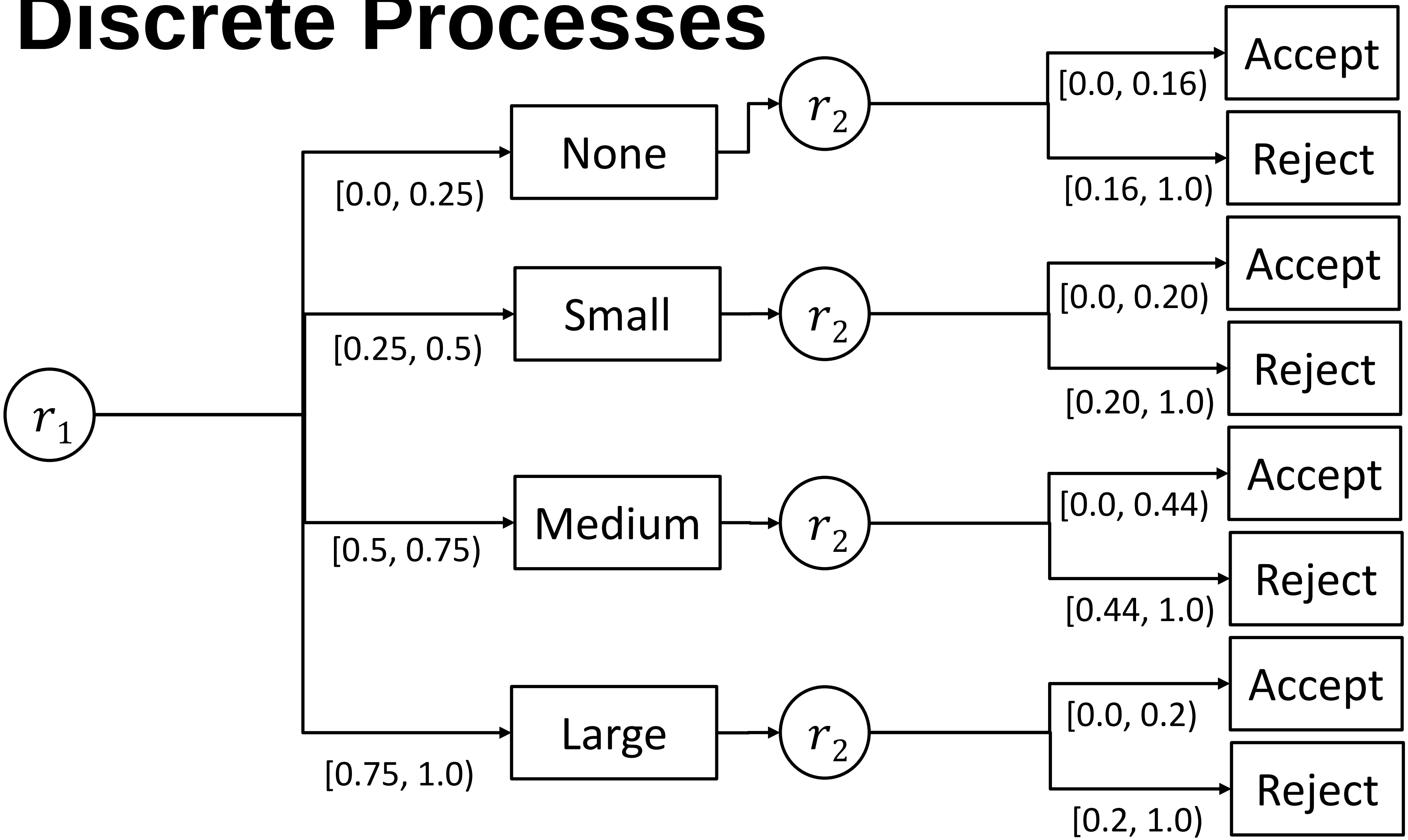


- Some CDFs are not easy to quantify or express
- Rely only on PMFs
- Example: Café Java



Copyright © 2018 Paul T. Grogan

Accept-Reject for Discrete Processes



Copyright © 2018 Paul T. Grogan

Continuous Process Generators

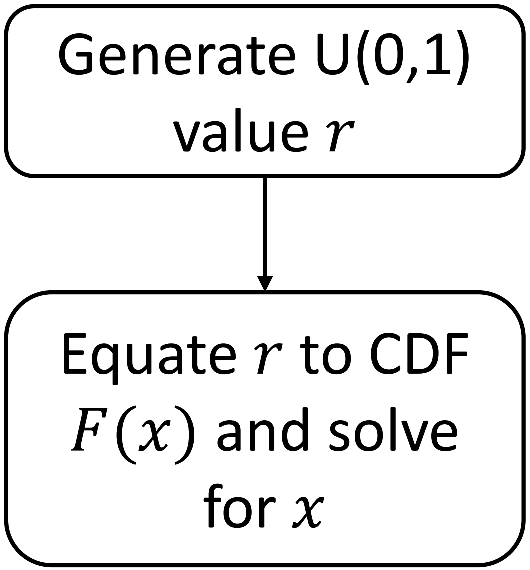
Continuous Process Generators

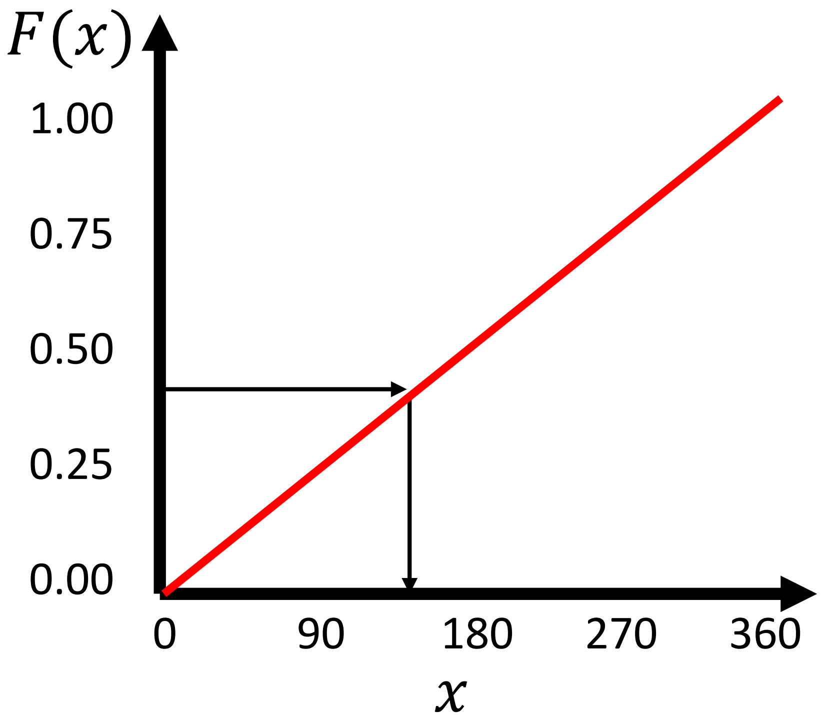
Continuous processes are a sequence of continuous random variable values

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

Copyright © 2018 Paul T. Grogan

Inverse Transform Method



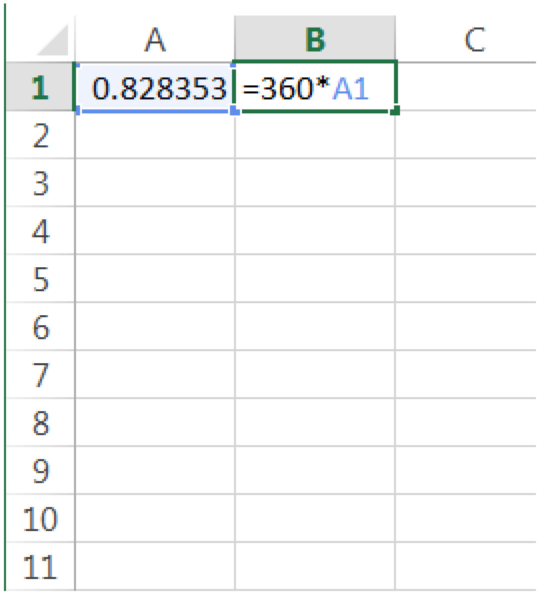


$$r = F(x) = \frac{x}{360}$$

$$\rightarrow x = 360 * r$$

Inverse Transform for Continuous Processes

```
import numpy as np

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



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

Copyright © 2018 Paul T. Grogan

Class Problem: Café Java VIP

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

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

$$f(x) = \lambda e^{-\lambda x}$$

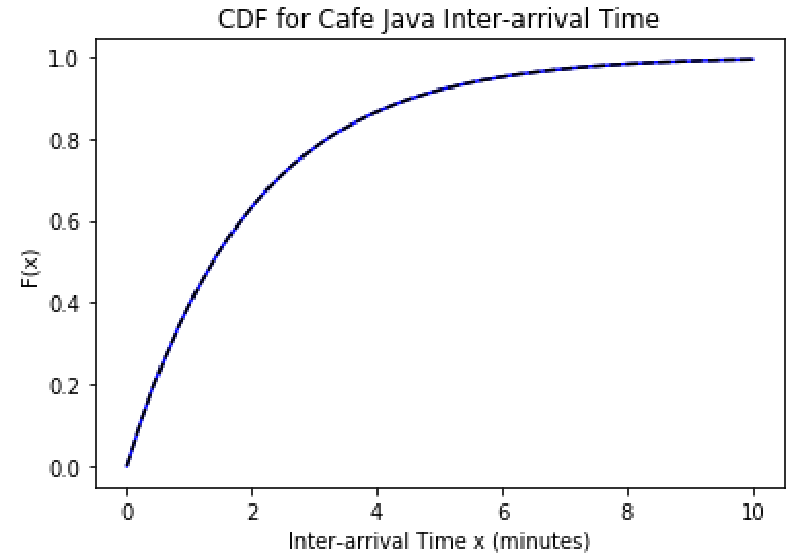
$$F(x) = \int_{i=0}^{x} f(i)\,di = \int_{i=0}^{x} \lambda e^{-\lambda i}\,di = 1 - e^{-\lambda x}$$

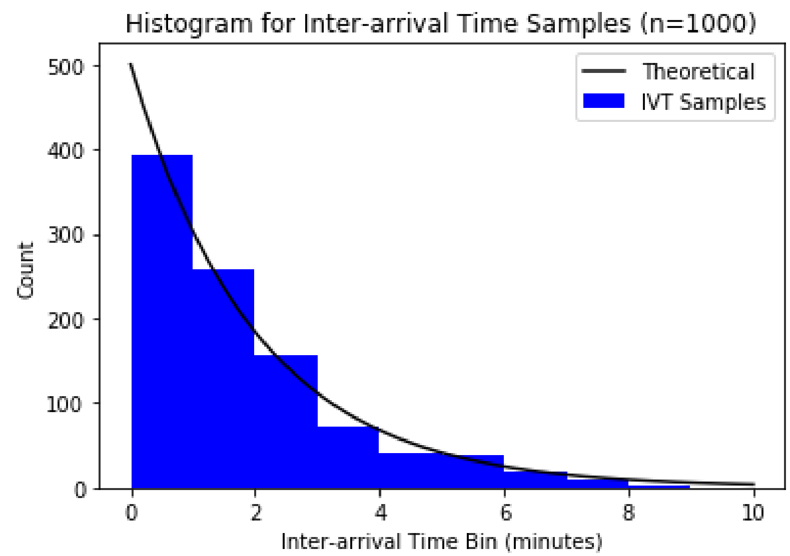
$$\rightarrow x = \frac{-\ln(1-r)}{\lambda}$$

Copyright © 2018 Paul T. Grogan

Café Java Arrival Time Generators

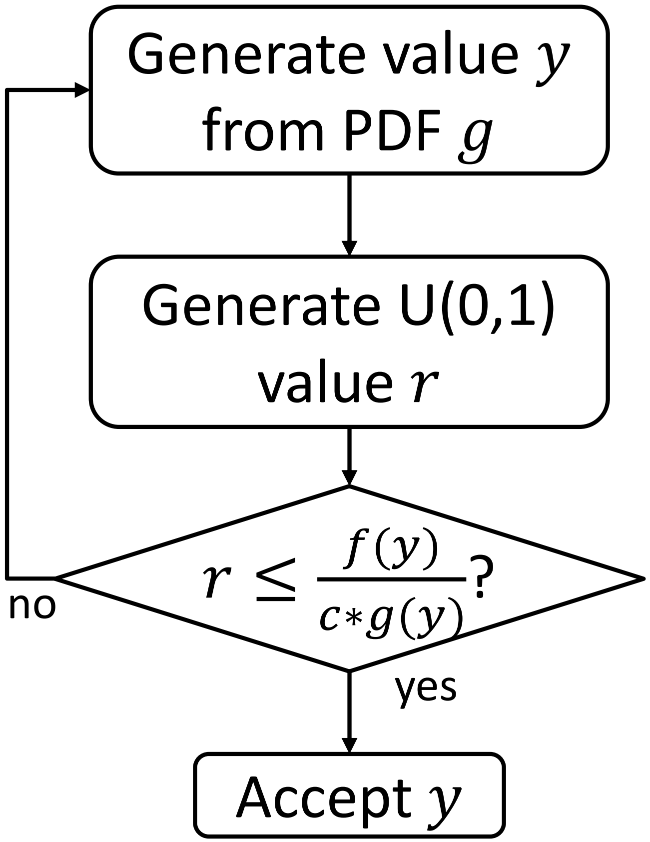
```
import numpy as np

_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)]
```





Copyright © 2018 Paul T. Grogan

Accept-Reject Method

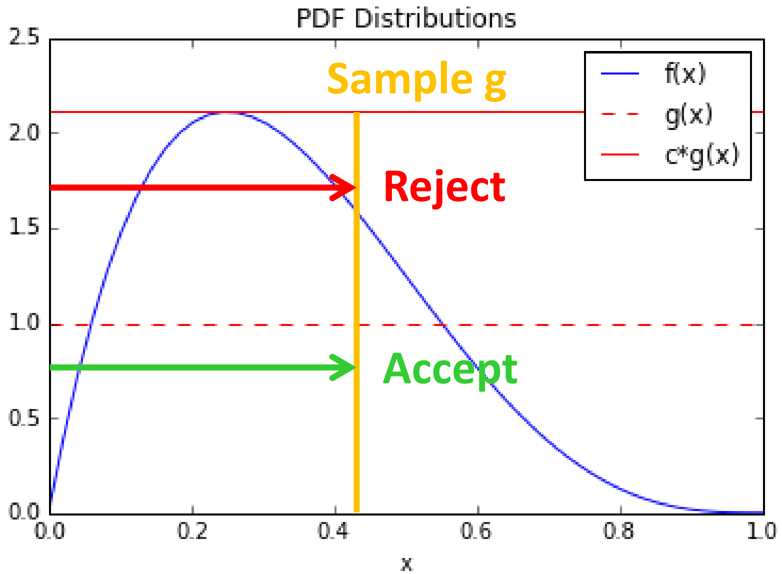


- Some CDFs do not have closed-form equations
- Rely only on PDFs
  - Use a simpler "enveloping" distribution $g(x)$ where $c * g(x) \geq f(x) \forall x$
  - Simplest: $g(x) \sim \text{uniform}(a, b)$
  - Find maximum $f(x)$ and assign c appropriately

Copyright © 2018 Paul T. Grogan

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?
  - $0 = f'(x) = 20(1-x)^3 - 60x(1-x)^2$
  - $\quad = -20(x-1)^2(4x-1)$
  - $\rightarrow f(0.25) = \frac{135}{64} \rightarrow c = \frac{135}{64}$
- $r \leq \frac{f(y)}{cg(y)} = \frac{256}{27} y(1-y)^3$
- Equivalently: $r * c \leq f(y)$



Copyright © 2018 Paul T. Grogan

Accept-Reject for Continuous Processes