Stochasticity

EES 4760/5760

Agent-Based and Individual-Based Computational Modeling

Jonathan Gilligan

Class #18: Thursday, October 28 2021

Stochastic Business Investor model

On “downloads” page

https://ees4760.jgilligan.org/models/class_18/class_18_models.zip

Stochasticity

Stochasticity:
Why do we use random numbers?

- To “inject ignorance” into a model:
  - We want to represent some kind of variability *but*
  - We do not want all the details of what causes the variability

```
ask patches [set profit 1000 + (random 1000)]
ask turtles [ if random-float 1.0 < mortality-prob [die]
]
```

Common uses of stochasticity

- Initialization

```
set fish-length random-normal 50 10
```

  Pick a random number from a normal distribution with mean 50 and standard deviation 10

- In submodels

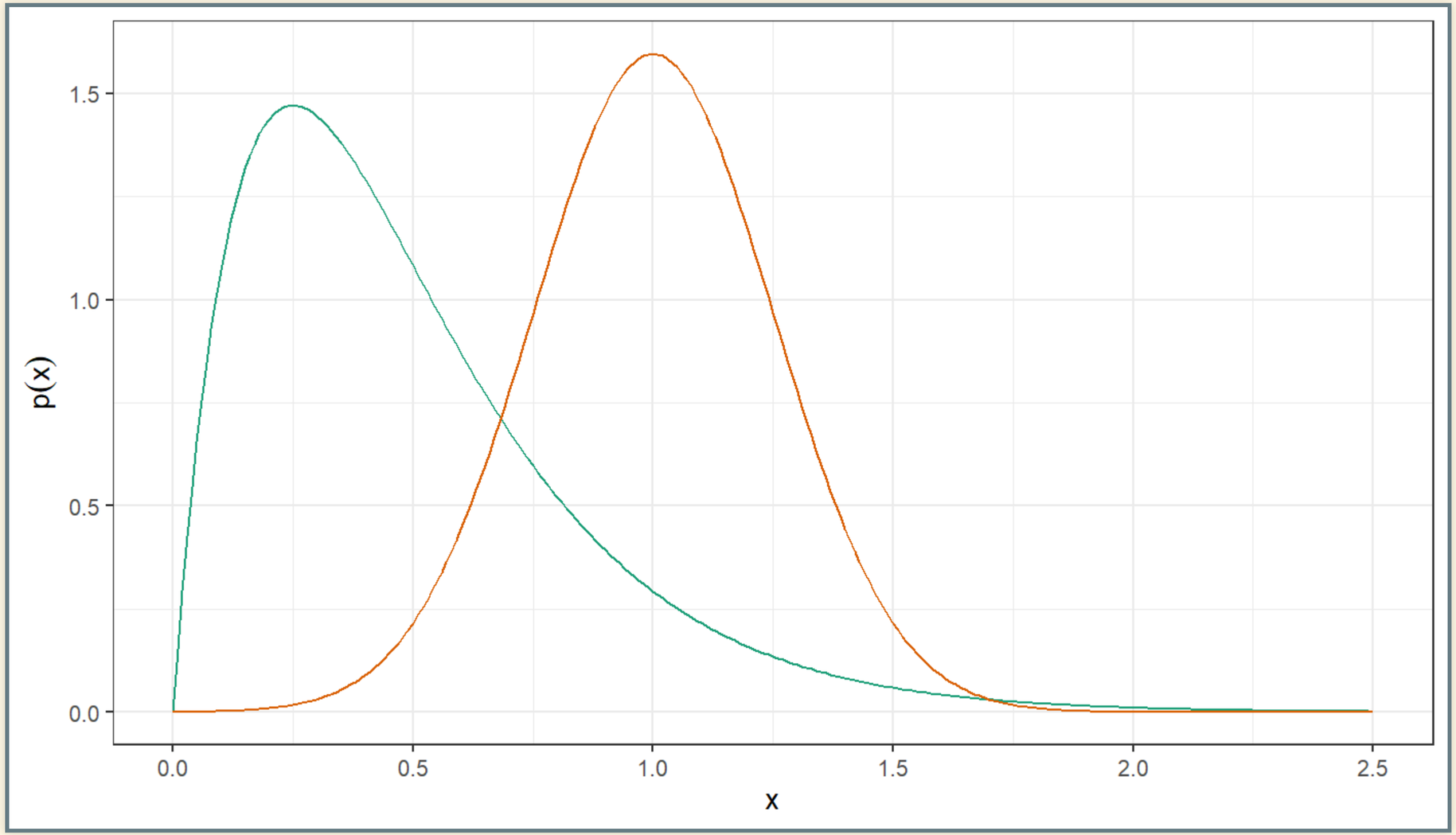
```
ifelse random-float 1.0 < q
[ uphill elevation ]
[ move-to one-of neighbors ]
```

Guidance for Stochasticity

- **Do** use stochasticity to initialize model differently on different runs
  - Makes sure that effects you see are not *artifacts* of a specific initialization
- **Do** use stochasticity to simplify representation of very complex processes
  - If wild dogs live an average of 5 years:
    - instead of a detailed submodel that determines exactly when each dog will die,
    - let dogs die at random with a 20% probability of dying each tick.
- **Don’t** use too much stochasticity
  - If you put too many different sources of randomness into your models every run may be *so different* you can’t discover any general properties.

Distributions

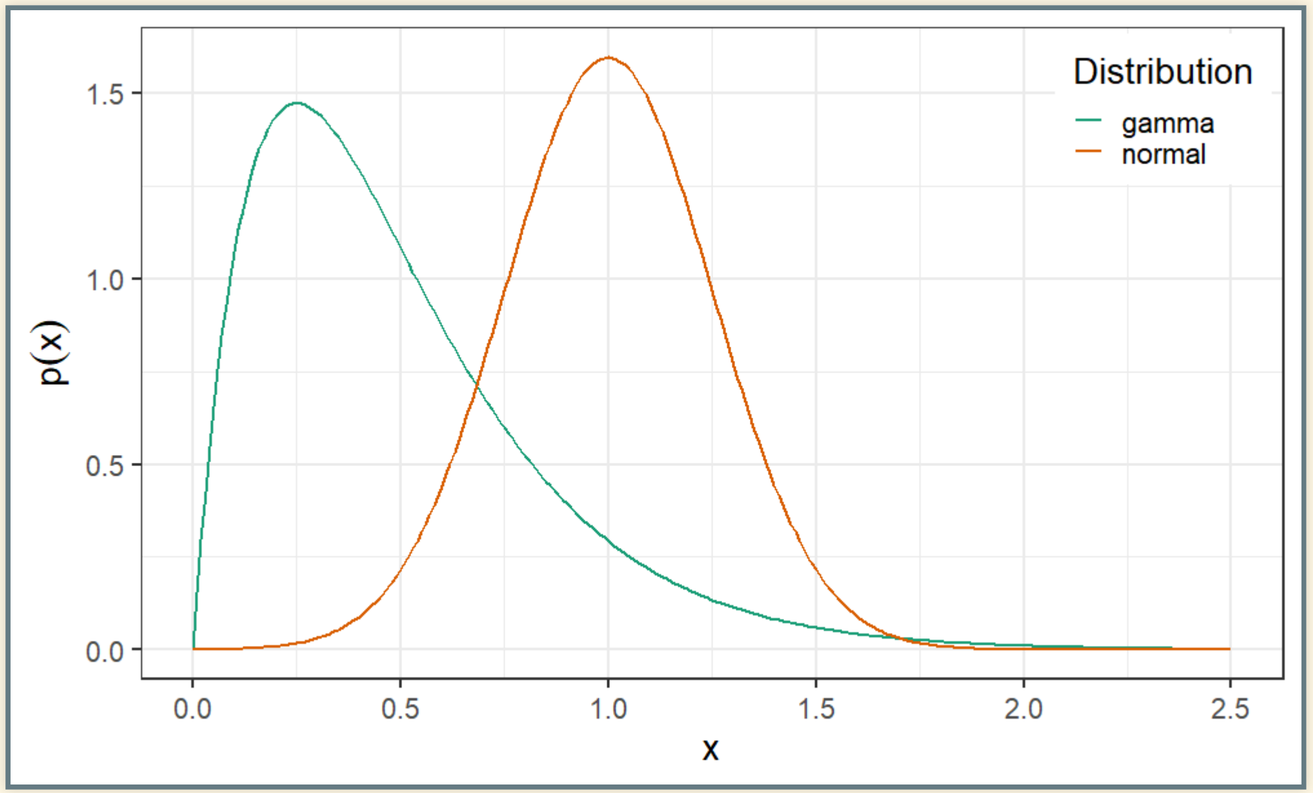
What is a Distribution?



What is a Distribution?

- In simulation programming, an algorithm that produces (pseudo)random numbers that fit a particular statistical distribution.

```
let x1 random-normal 1.0  0.25
let x2 random-gamma  2.0  4.0
```



Distributions in NetLogo

- Continuous (real-number)
  - Uniform: `random-float` *`upper-limit`*
  - Normal: `random-normal` *`mean sd`* (beware of outliers)
  - Also: `random-gamma`, `random-exponential`
- Discrete (integer):
  - Uniform: `random` *`upper-limit`*
    - 0 to *`upper-limit - 1`*
  - Poisson: `random-poisson` *`mean`*
    - `mean` = average value
  - Bernoulli (`true` or `false`): `random-float 1.0 < p`
    - `true` with probability `p`
    - See `random-bernoulli` reporter on p. 200 of the textbook.

Controlling randomness

Controlling randomness

- `random-seed` *number*
  - As long as *number* is the same, you get the same sequence of random numbers

```
to setup
    clear-all
    random-seed 12345
    ...
end
```

Controlling randomness

- `with-local-randomness [ commands ]`

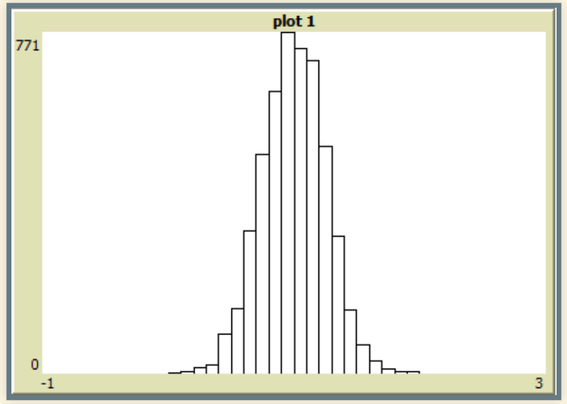
  Runs without changing sequence of random numbers in other parts of the model

```
to move
  with-local-randomness
  [
    random-seed 565656
    ...
  ]
end
```

How can we see a distribution?

- Histograms

```
to plot-histogram-normal
  clear-all
  set-plot-pen-mode 1 ; bar mode
  set-plot-pen-interval 0.1
  set-plot-x-range -1 3
  let x (list)
  ; fill x with 5000 random numbers from a normal distribution
  repeat 5000 [ set x fput (random-normal 1.0 0.25) x]
  histogram x
end
```



Uniform distributions

- Integer: `random n` gives an integer i: $0 \leq i < n$
  - From 0 to $(n - 1)$
- Continuous: `random-float z` gives a number x: $0 \leq x < z$
  - Should we worry that $x < z$?

```
to test
  let num_draws 10000
  let max-rand 0
  repeat num_draws
  [
    let x random-float 1000
    if x > max-rand [ set max-rand x ]
  ]
  show max-rand
end
```

```
observer> test
observer: 999.9869678378017
```

Poisson distribution

- For countable things that happen at a small rate.
  - On every turn a random number of agents turn red, with an average of 5% of agents

```
ask n-of (random-poisson (0.05 * count turtles)) turtles
[set color red]
```

or

```
let n-red random-poisson (0.05 * count turtles)
ask n-of n-red turtles [set color red]
```

Normal distribution

- For measurable things with an average value

```
set weight random-normal 150 20  ; weight in pounds
set height random-normal 70 2    ; height in inches
```

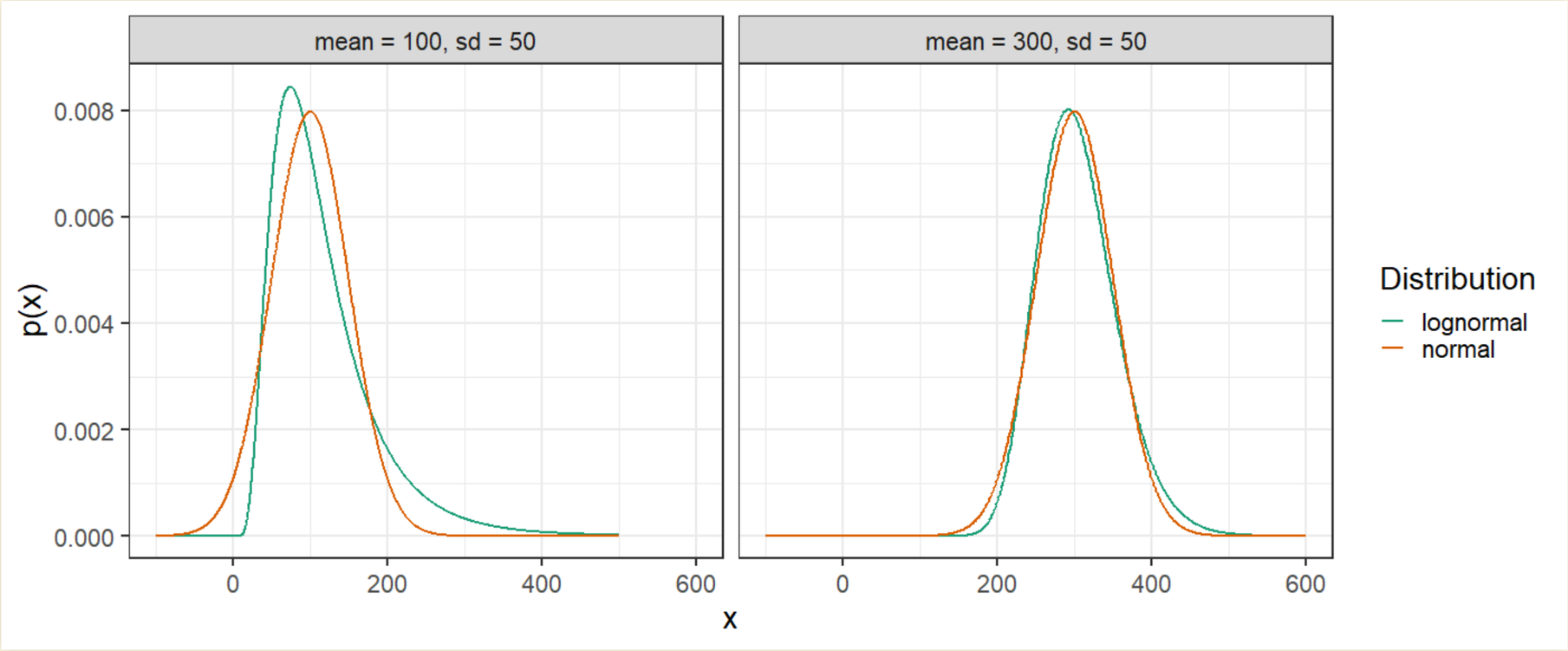
- Be careful of outliers. There is no limit, so there is a small probability of getting a very large value or a negative value.

```
repeat 5000 [
  let x random-normal 30 10
  if x < 0 [ print x ]
]
```

Lognormal distribution

- If you want something like a normal distribution, but where the result **must** be positive, try a lognormal distribution:

```
to-report random-lognormal [ m s ]
  let mm ln m
  let ss (ln ((m + s) / (m - s))) / 2
  report exp (random-normal mm ss)
end
```



Stochastic Business Investors

Stochastic Business Investors

Model: https://ees4760.jgilligan.org/models/class_18/business_investor_class_18.nlogo

Original model:

- Investors move to neighbor with highest expected utility (including own patch)
- Average over 10,000 runs:

Alternative	Frequency
Higher profit, lower risk	78%
Higher profit, higher risk	9.3%
Lower profit, lower risk	3.4%
Lower profit, higher risk	0%
Don’t move	92.1%

- Mean wealth = $212,434
- Total wealth = $5,310,861

Stochastic Model

Original model:

Alternative	Frequency
Higher profit, lower risk	78%
Higher profit, higher risk	9.3%
Lower profit, lower risk	3.4%
Lower profit, higher risk	0%
Don’t move	92.1%

Stochastic model

- If there are neighbors with higher profit and lower risk:
  - 78% probability of moving to one of them
- Otherwise, if there are neighbors with higher profit and higher risk:
  - 9.3% probability of moving to one of them
- etc.

Compare models: