MAIN CONCEPTS OF SIMULATION

Dr. Aric LaBarr

Institute for Advanced Analytics

SIMULATION INTRODUCTION

Varying Inputs

- Up until this point we have been assuming a rather unrealistic view of the real world – **certainty**.
- In a real-world setting – especially the business world – the inputs and coefficients in a problem are rarely fixed quantities.
- Optimization techniques like sensitivity analysis – **reduced cost** and **shadow prices** – are one approach to handling this problem.

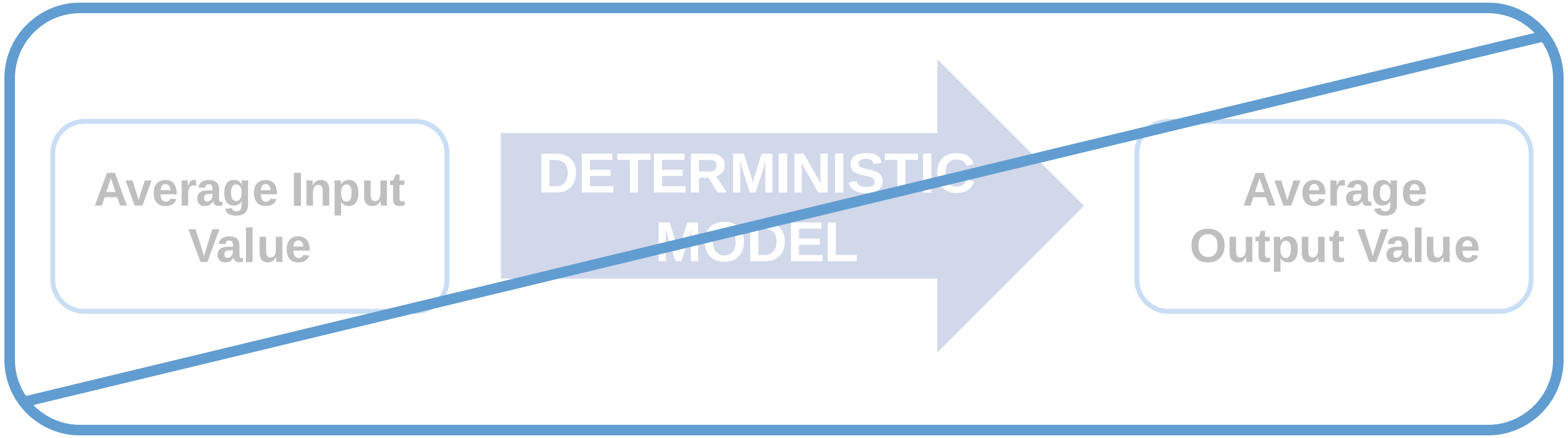
Monte Carlo Simulations

- Uncertainty is foundational in Monte Carlo simulations.

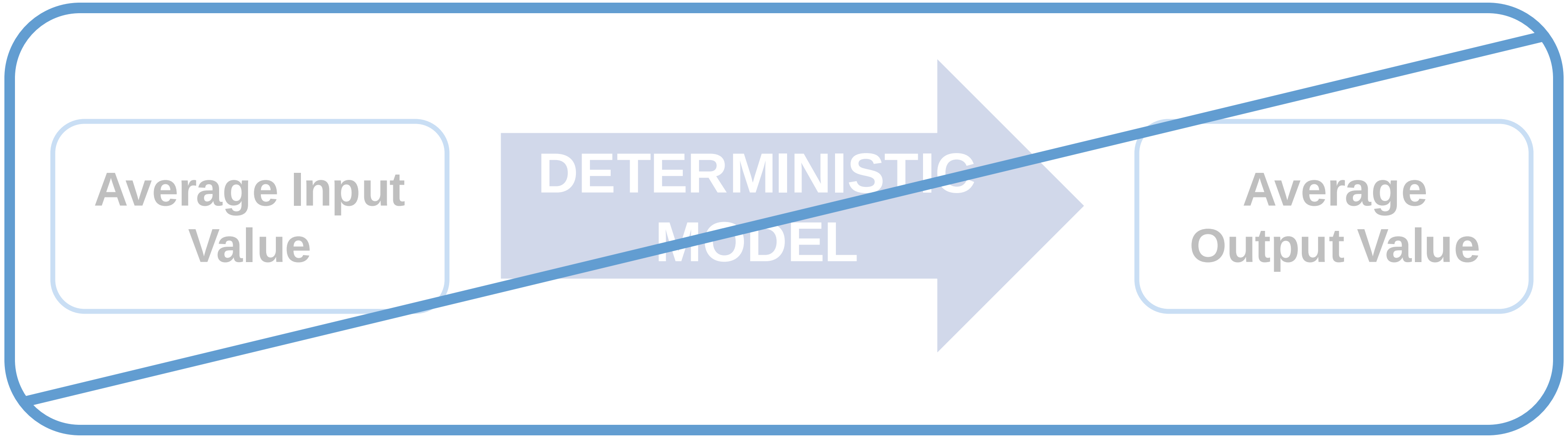
- **Simulations** help us determine not only the full array of outcomes of a given decision, but the probabilities of these outcomes occurring.
- Some examples:
  - Risk analysis – how rare certain outcomes actually are.
  - Model evaluation – how good is our model compared to others.

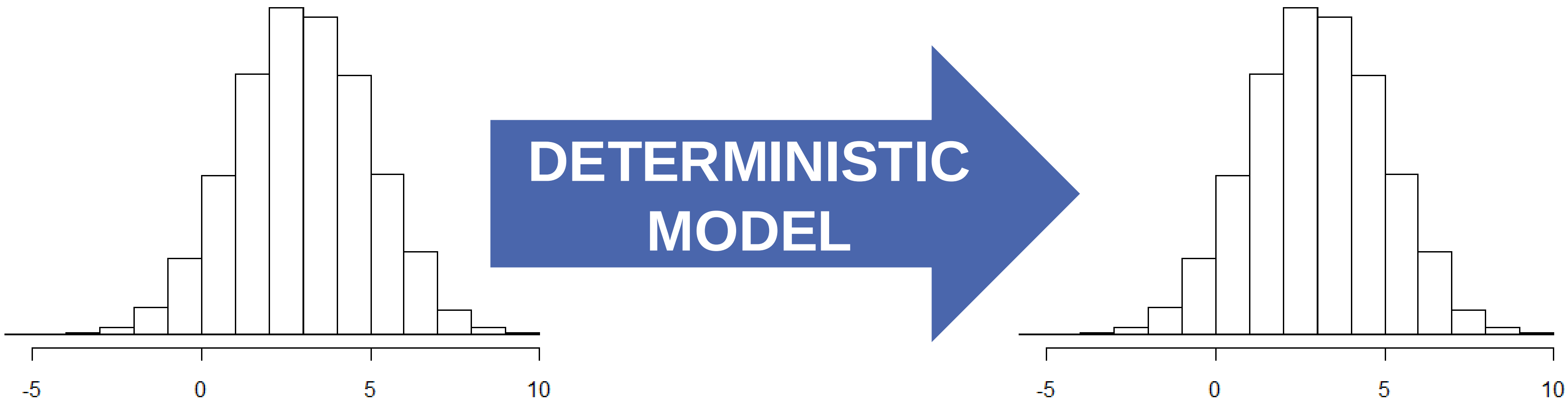
Monte Carlo Simulations

Monte Carlo Simulations



Monte Carlo Simulations



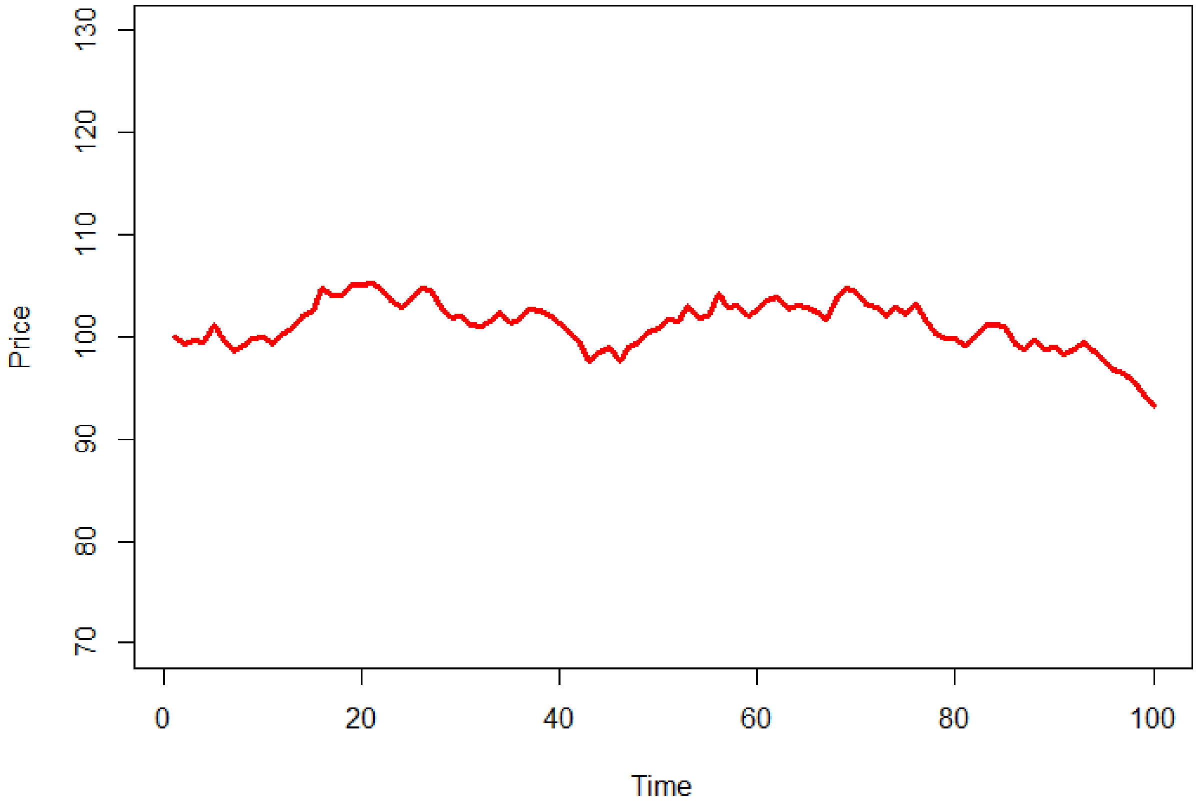


What-If Analysis

- Each input inside of a model (or process) is assigned a range of possible values – the **probability distribution of the inputs**.
- We then analyze what happens to the decision from our model (or process) under all of these possible scenarios.
- Simulation analysis describes not only the outcomes of certain decisions, but also the **probability distribution of those outcomes** – the probability each of these outcomes occurs.

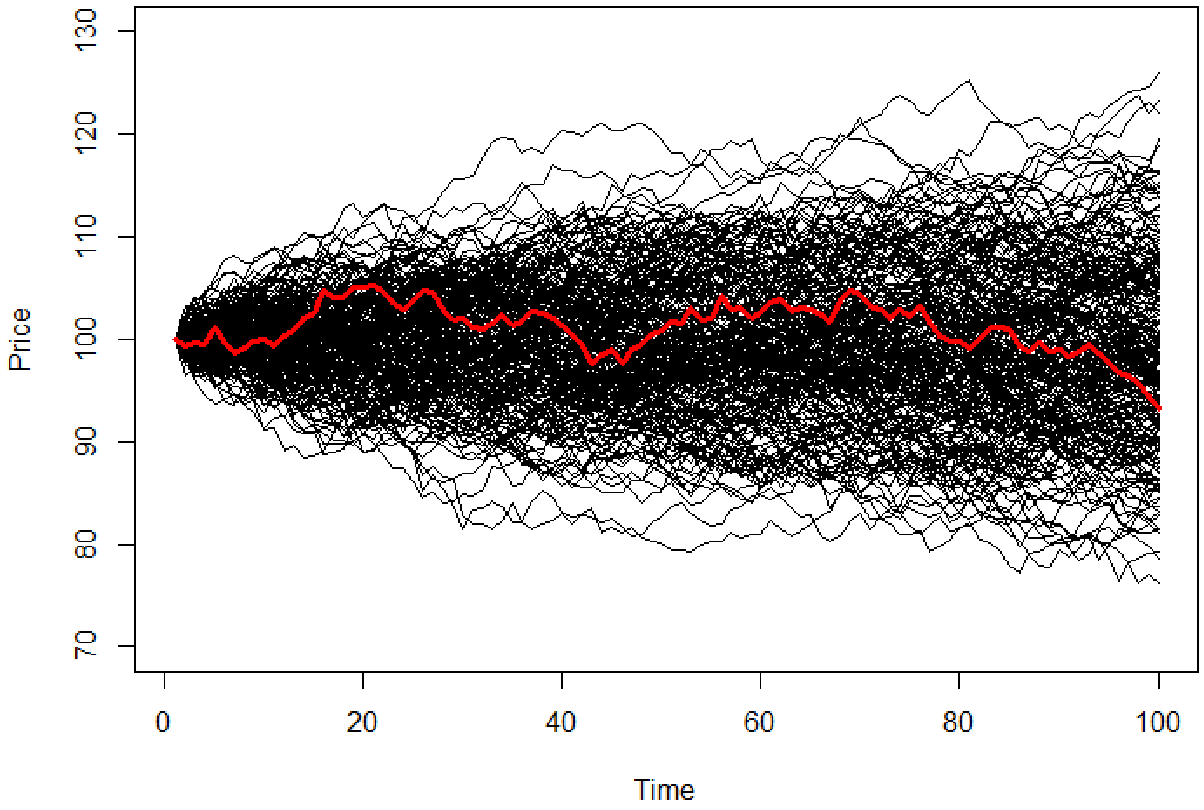
What-If Analysis

- Assume a stock price is $100.
- Follows a random walk for next 100 days with $\varepsilon_t \sim N(0,1)$.



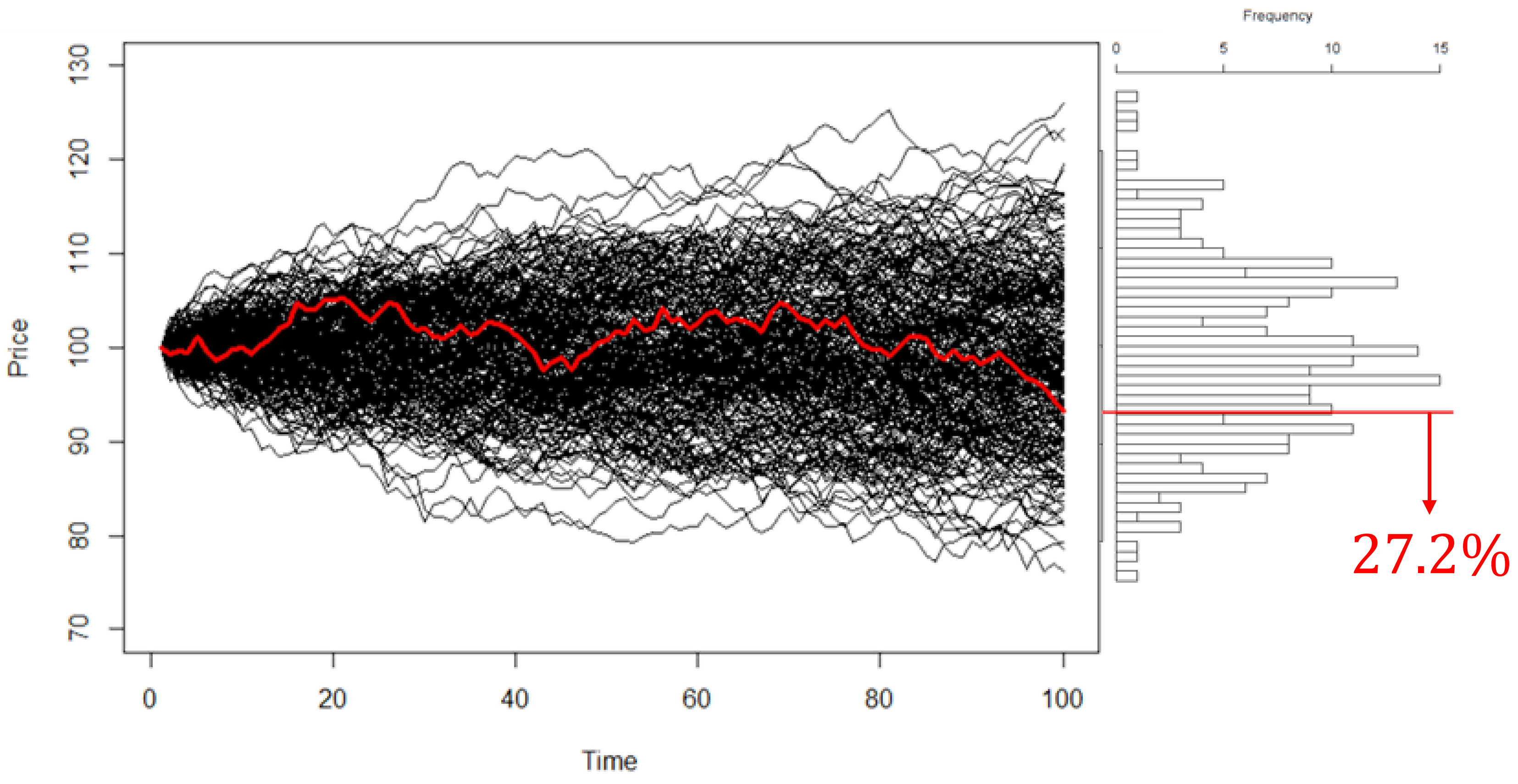
What-If Analysis

- Assume a stock price is $100.
- Follows a random walk for next 100 days with $\varepsilon_t \sim N(0,1)$.



What-If Analysis

- Assume a stock price is $100.
- Follows a random walk for next 100 days with $\varepsilon_t \sim N(0,1)$.



Outcome Distribution

- Simulation analysis describes not only the outcomes of certain decisions, but also the **probability distribution of those outcomes** – the probability each of these outcomes occurs.
- After the simulation analysis, the focus then turns to the probability distribution of the outcomes.
- Describe the characteristics of this new distribution – mean, variance, skewness, kurtosis, percentiles, etc.

Example

- You want to invest \$1,000 in the US stock market for one year.
- You invest in a mutual fund that tries to produce the same return as the S&P500 Index.

$$P_1 = P_0 + r_{0,1} * P_0$$

OR

$$P_1 = P_0 * \left(1 + r_{0,1}\right)$$

Example

- You want to invest $1,000 in the US stock market for one year.
- You invest in a mutual fund that tries to produce the same return as the S&P500 Index.

$$P_1 = P_0 + r_{0,1} * P_0$$

OR

$$P_1 = P_0 * (1 + r_{0,1})$$

Initial Investment

Return

Selecting Distributions

- When designing your simulations the biggest choice comes from the decision of the distribution on the inputs that vary.
- 3 Methods:
  1. Common Probability Distribution
  2. Historical (Empirical) Distribution
  3. Hypothesized Future Distribution

Example

- You want to invest \$1,000 in the US stock market for one year.
- You invest in a mutual fund that tries to produce the same return as the S&P500 Index.

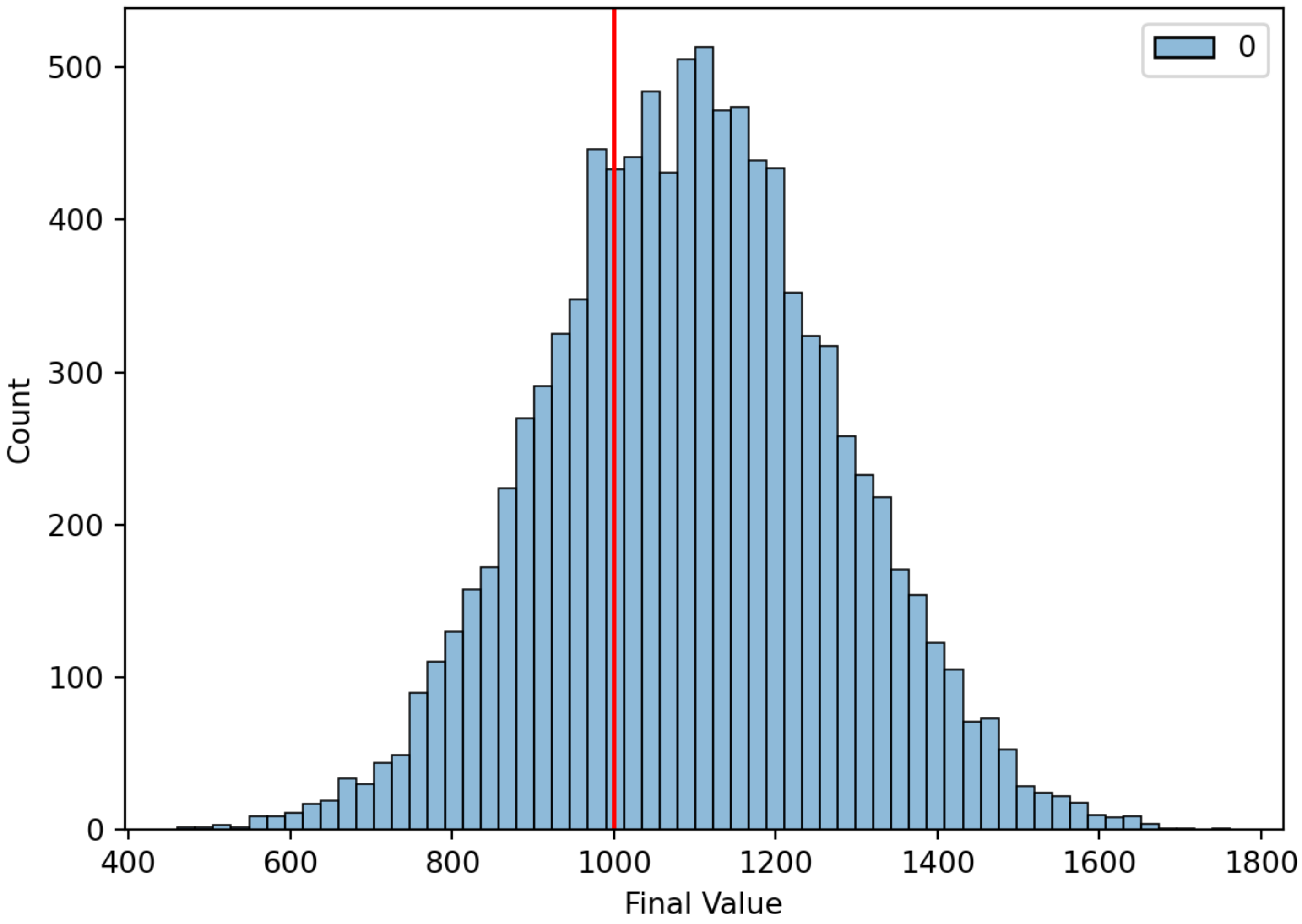
$$P_1 = P_0 * \left(1 + r_{0,1}\right)$$

- Assume annual returns follow a Normal distribution with historical mean of 9.75% and std. dev. of 18%.

Introduction to Simulation – Python

```
import pandas as pd
import numpy as np

np.random.seed(112358)

data = []

for i in range(10000):

  ret = np.random.normal(loc = 0.0975, scale = 0.18)

  P0 = 1000

  P1 = P0 * (1 + ret)

  data.append(P1)

df = pd.DataFrame(data)
```

Introduction to Simulation – Python



DISTRIBUTION SELECTION

Selecting Distributions

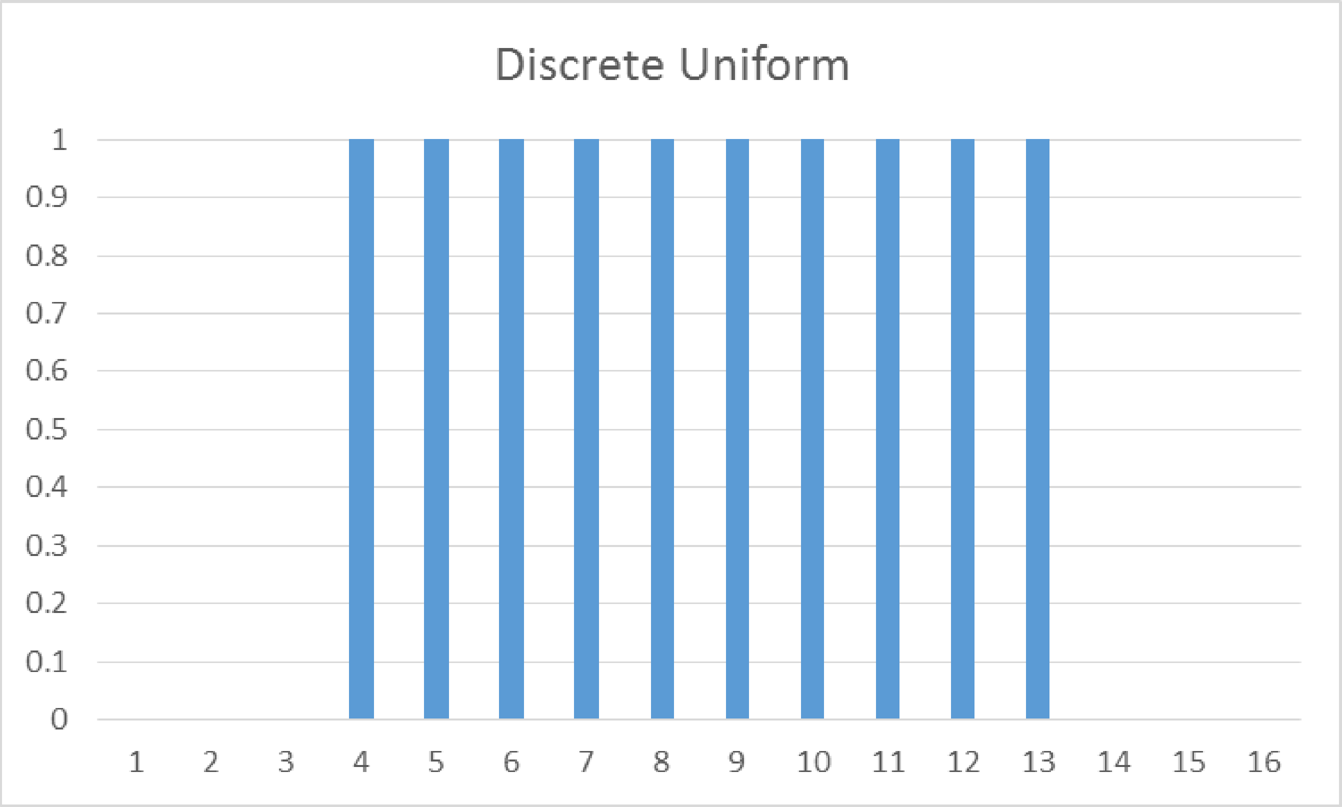
- When designing your simulations the biggest choice comes from the decision of the distribution on the inputs that vary.
- 3 Methods:
  1. Common Probability Distribution
  2. Historical (Empirical) Distribution
  3. Hypothesized Future Distribution

Common Probability Distributions

- Typically, we assume a common probability distribution for inputs that vary in a simulation.
- Common Discrete Distributions:
  1. Uniform Distribution
  2. Poisson Distribution

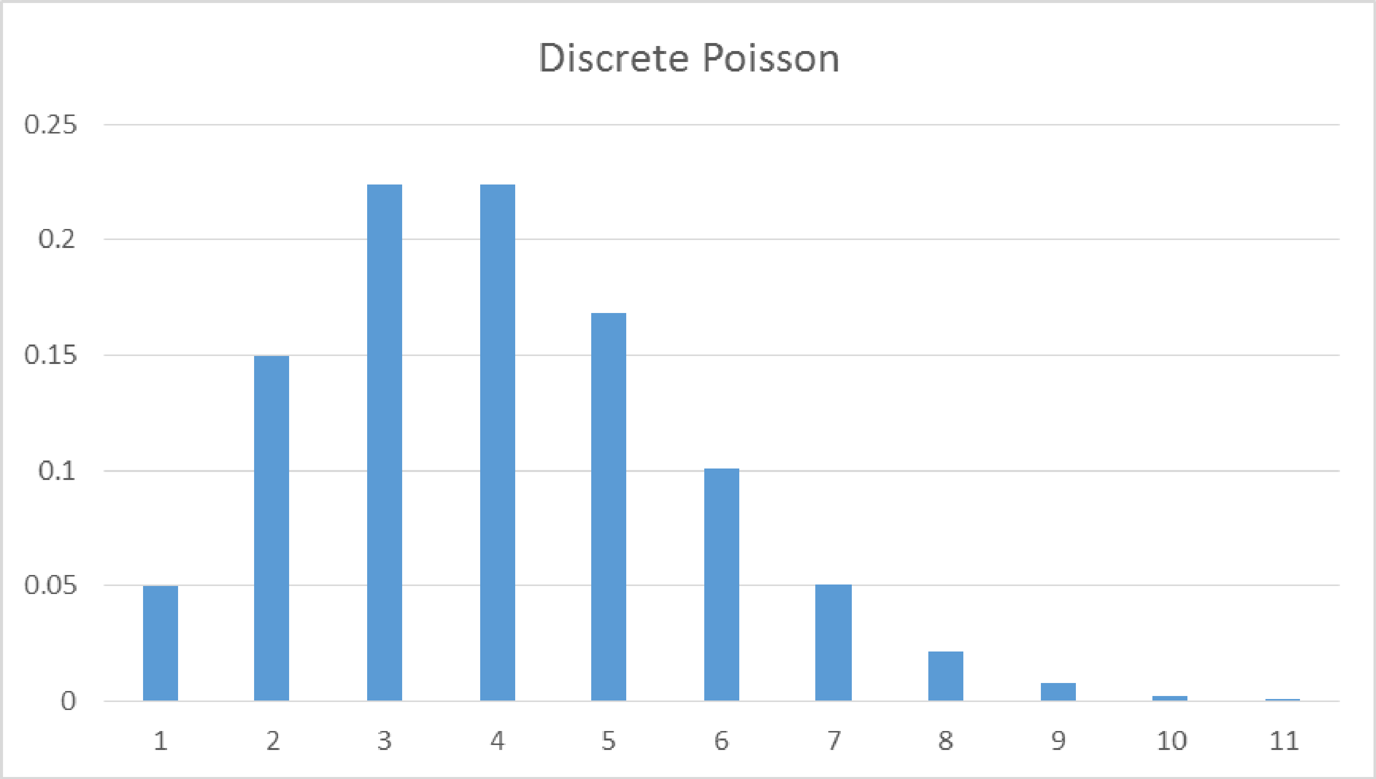
Common Probability Distributions

- Typically, we assume a common probability distribution for inputs that vary in a simulation.
- Common Discrete Distributions:



Common Probability Distributions

- Typically, we assume a common probability distribution for inputs that vary in a simulation.
- Common Discrete Distributions:

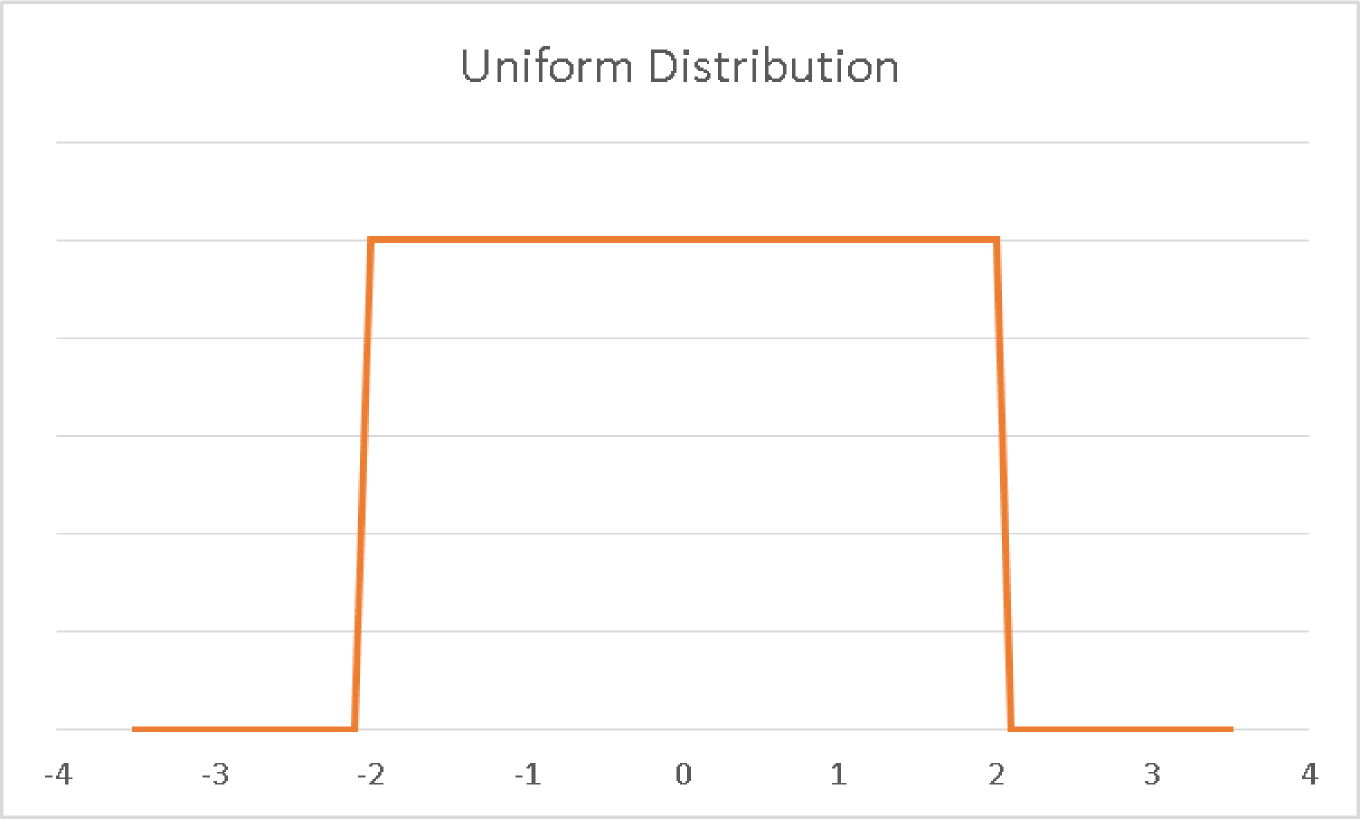


Common Probability Distributions

- Typically, we assume a common probability distribution for inputs that vary in a simulation.
- Common Continuous Distributions:
  1. Continuous Uniform Distribution
  2. Triangular Distribution
  3. Student's t-Distribution
  4. Lognormal Distribution
  5. Normal Distribution
  6. Exponential Distribution
  7. Chi-Square Distribution
  8. Beta Distribution

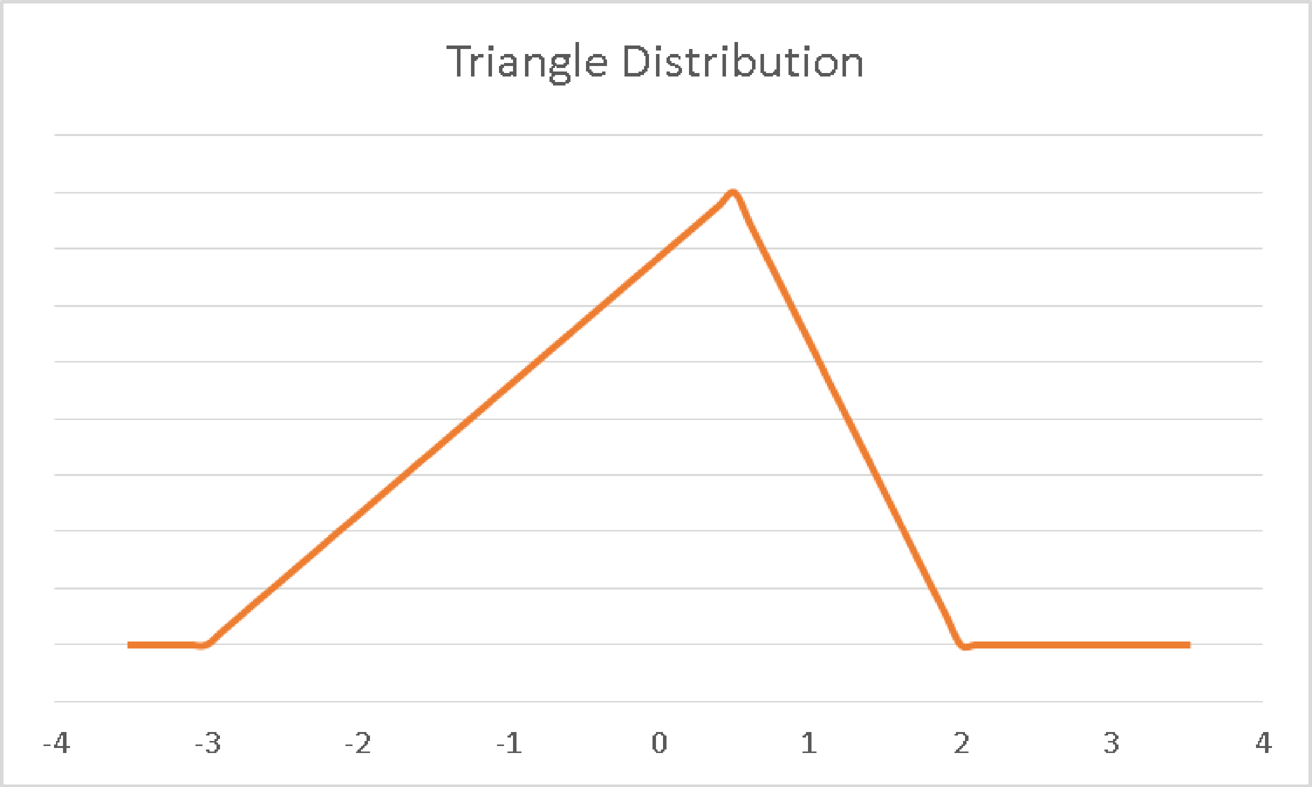
Common Probability Distributions

- Typically, we assume a common probability distribution for inputs that vary in a simulation.
- Common Continuous Distributions:



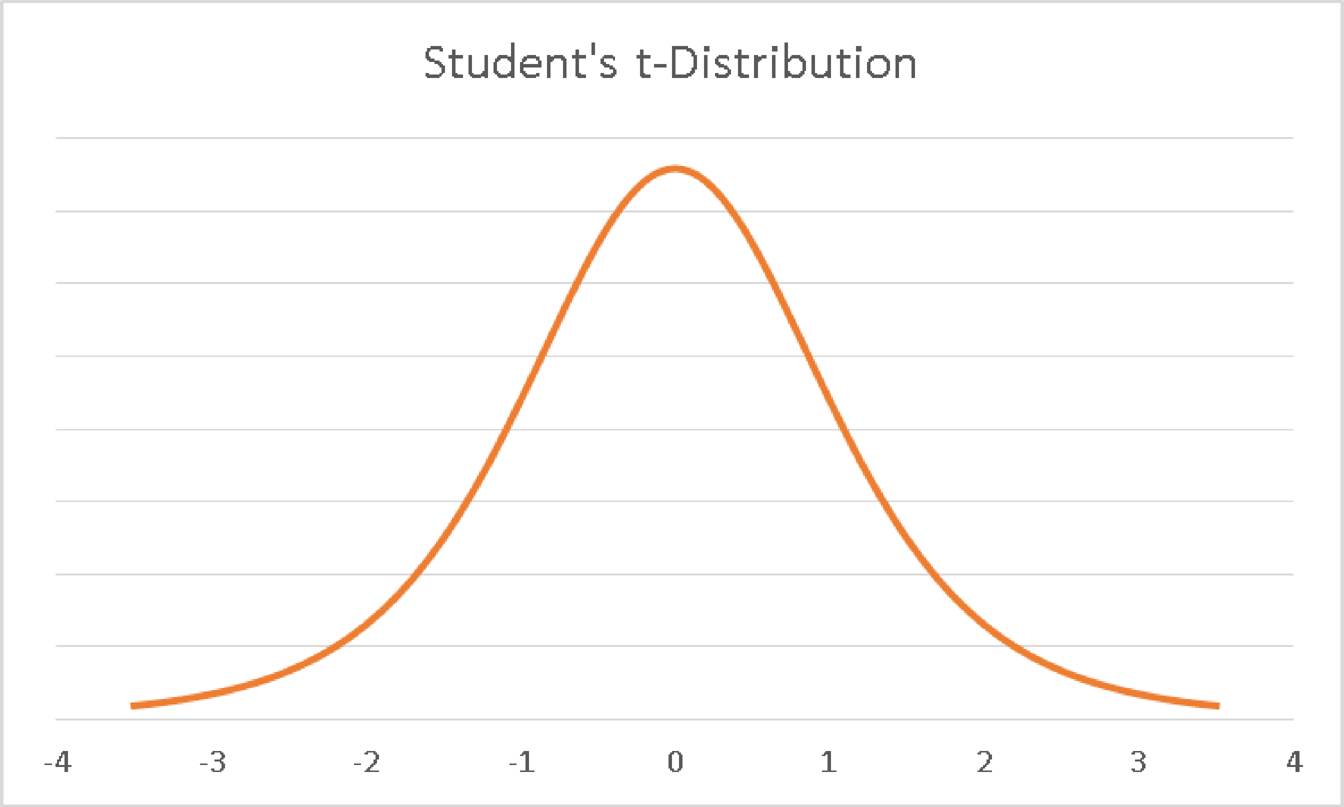
Common Probability Distributions

- Typically, we assume a common probability distribution for inputs that vary in a simulation.
- Common Continuous Distributions:



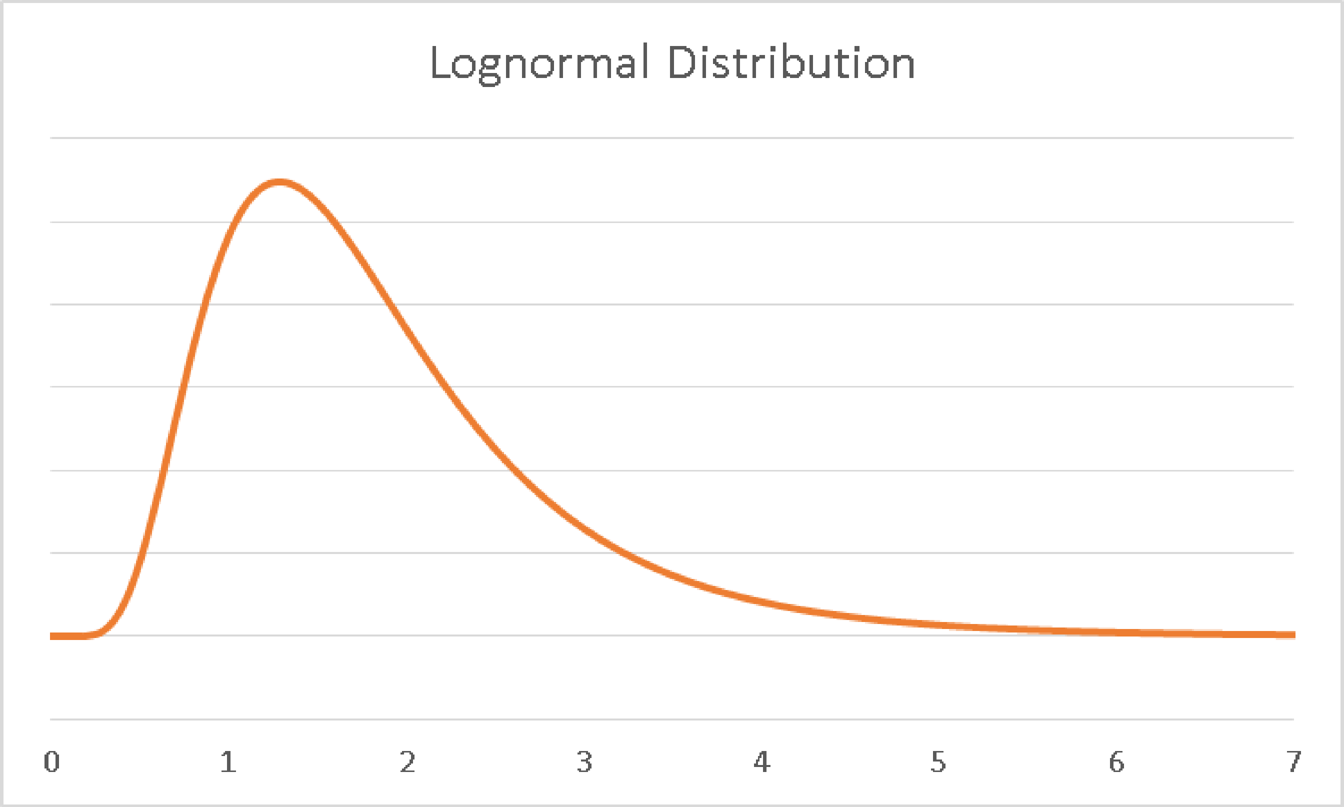
Common Probability Distributions

- Typically, we assume a common probability distribution for inputs that vary in a simulation.
- Common Continuous Distributions:



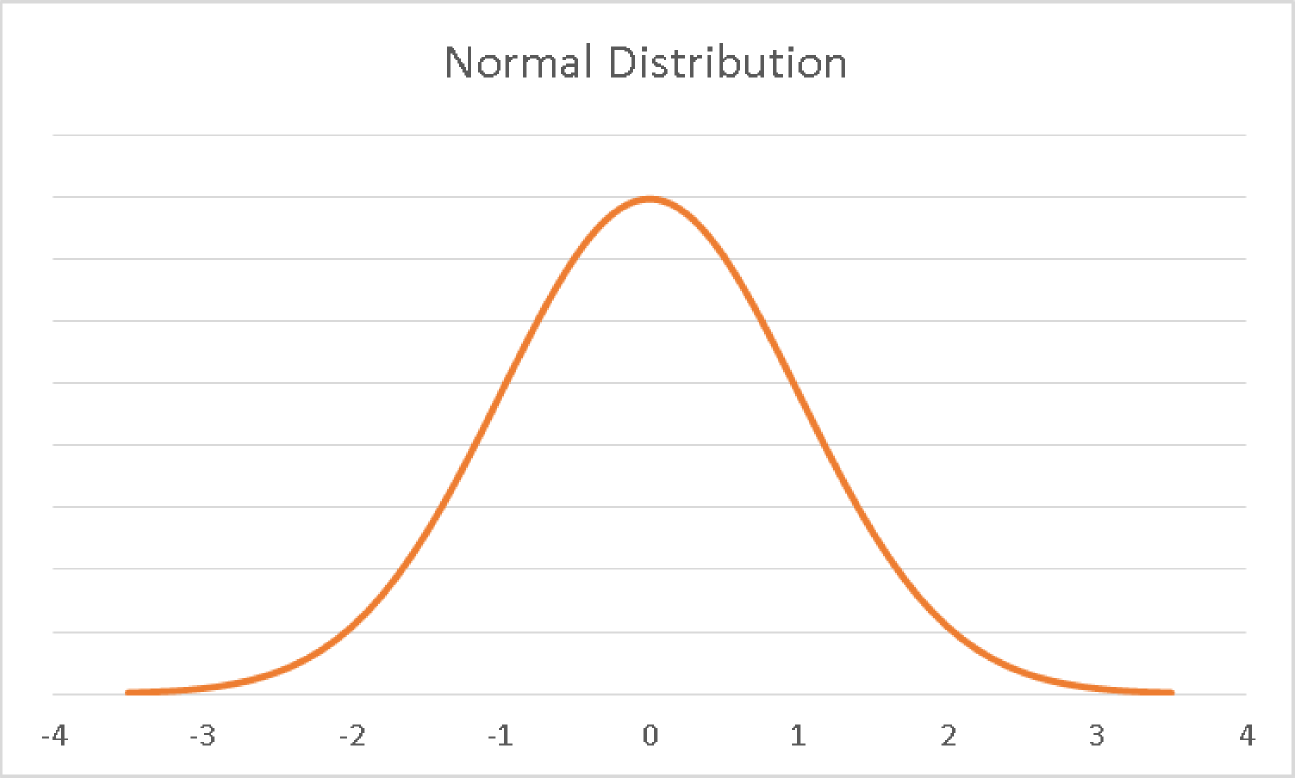
Common Probability Distributions

- Typically, we assume a common probability distribution for inputs that vary in a simulation.
- Common Continuous Distributions:



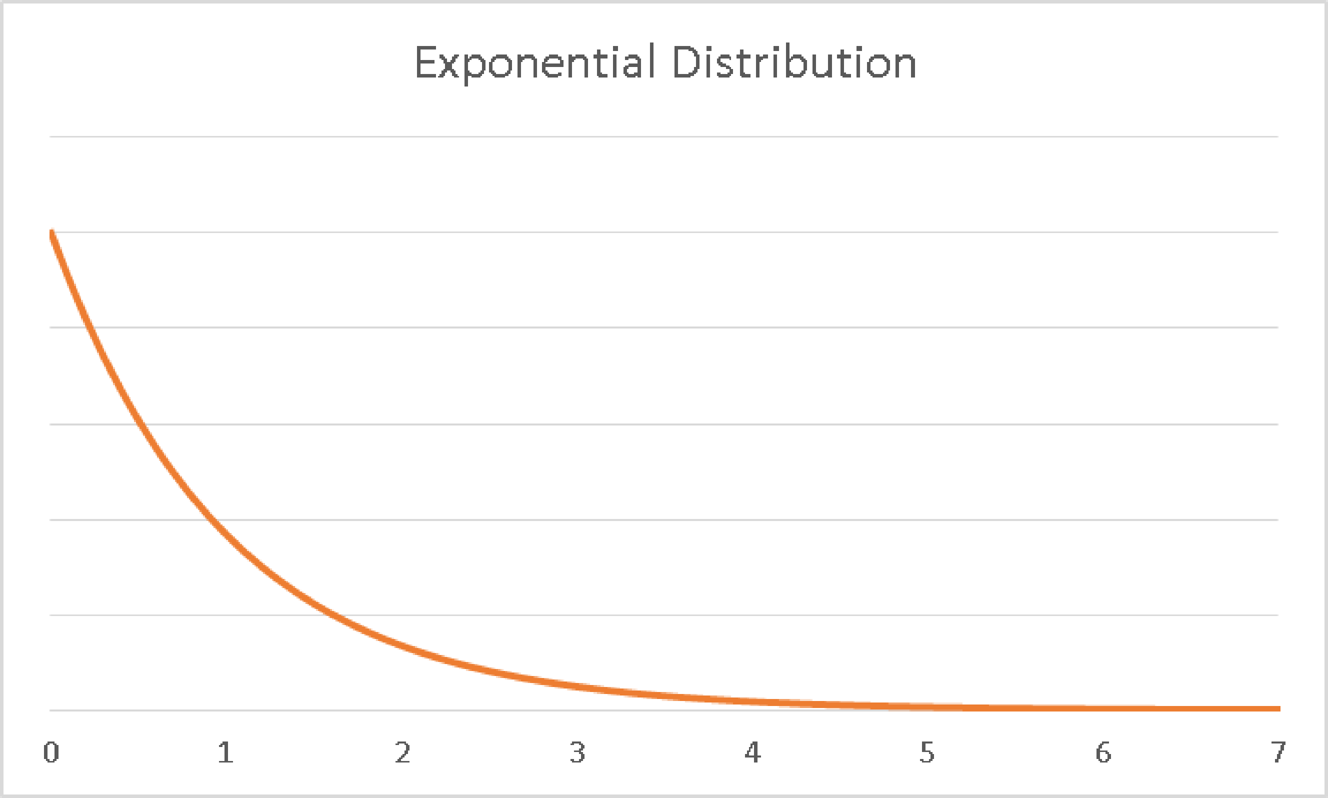
Common Probability Distributions

- Typically, we assume a common probability distribution for inputs that vary in a simulation.
- Common Continuous Distributions:



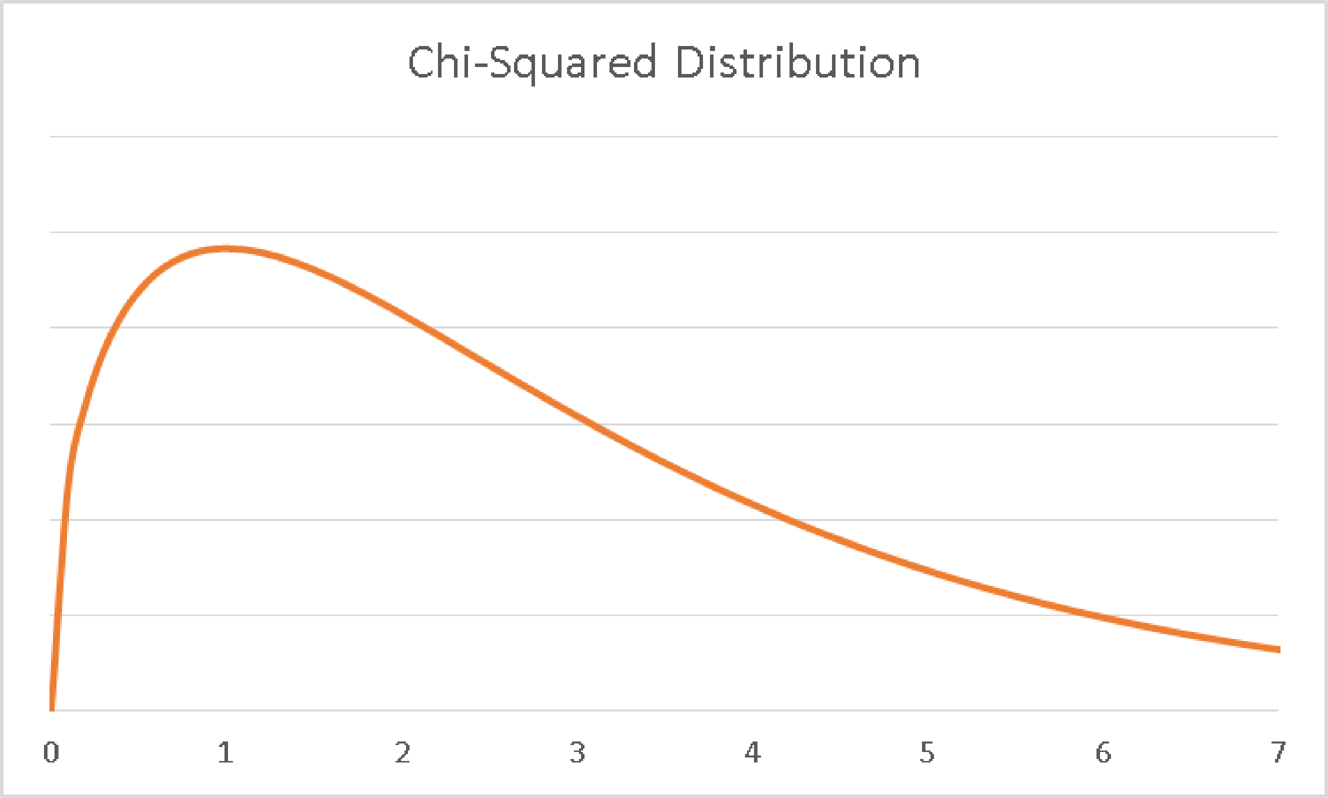
Common Probability Distributions

- Typically, we assume a common probability distribution for inputs that vary in a simulation.
- Common Continuous Distributions:



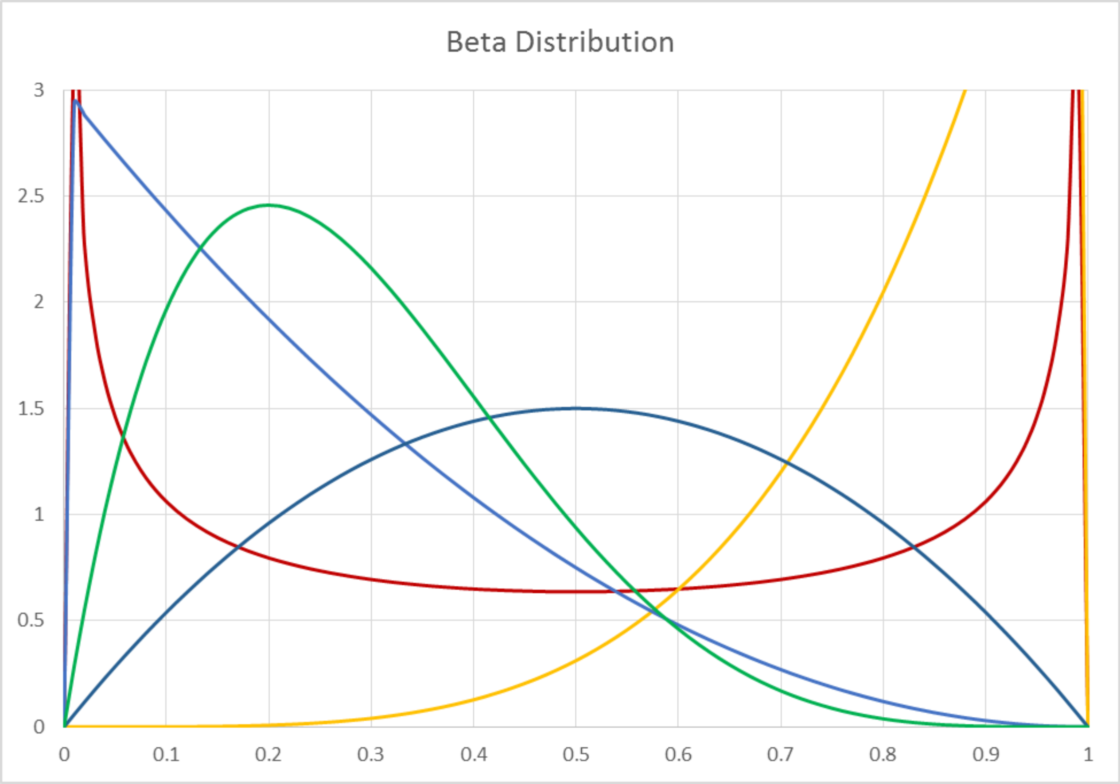
Common Probability Distributions

- Typically, we assume a common probability distribution for inputs that vary in a simulation.
- Common Continuous Distributions:



Common Probability Distributions

- Typically, we assume a common probability distribution for inputs that vary in a simulation.
- Common Continuous Distributions:



Historical (Empirical) Distributions

- If you are unsure of the distribution of the data you are trying to simulate, you can estimate it using **kernel density estimation**.
- Kernel density estimation is a non-parametric method of estimating distributions of data through smoothing out of data values.

Historical (Empirical) Distributions

- The Kernel density estimator is as follows:

$$\hat{f}(x) = \frac{1}{nh}\sum_{i=1}^{n} \kappa\left(\frac{x - x_i}{h}\right)$$

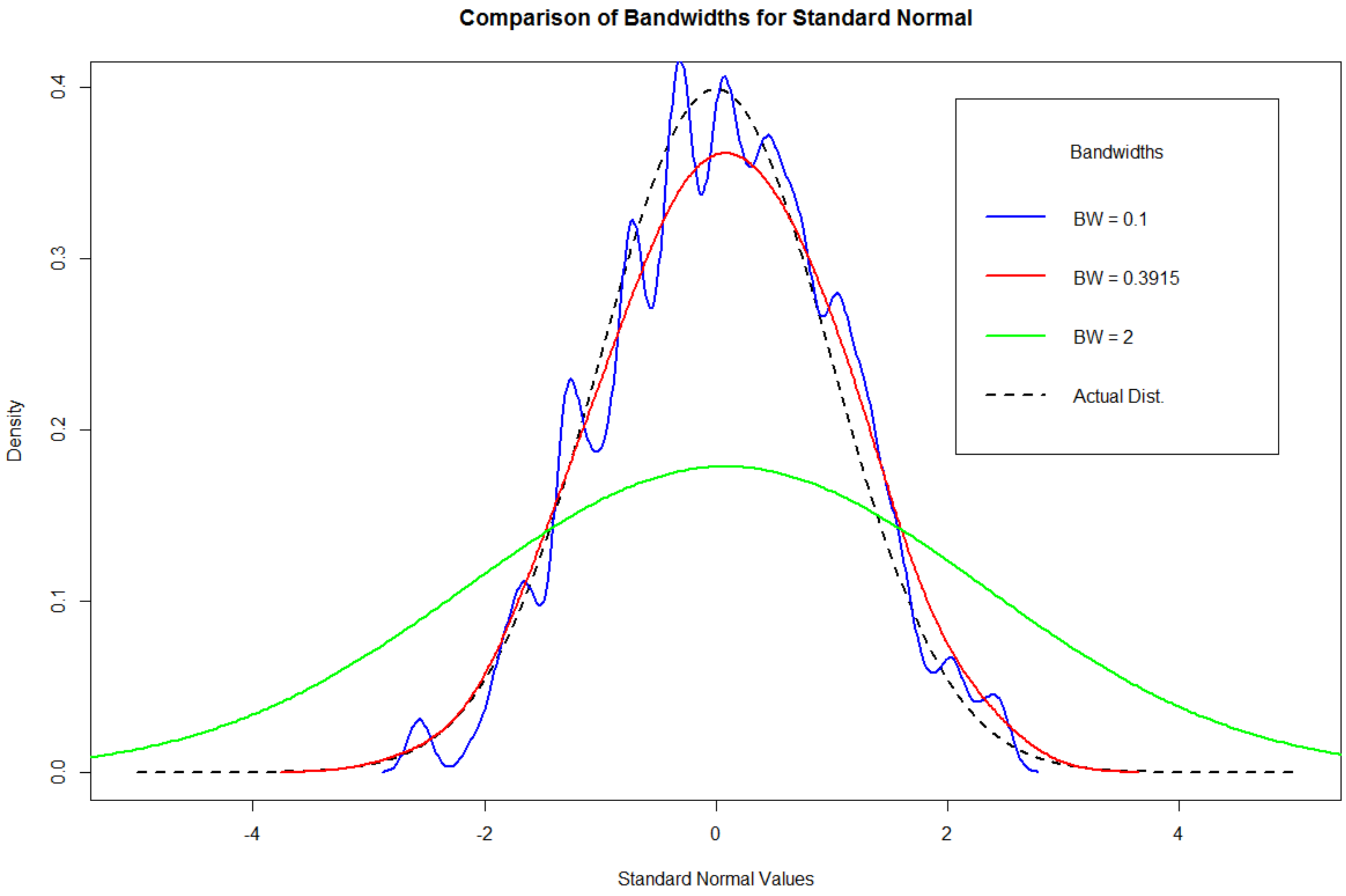
Historical (Empirical) Distributions

- The Kernel density estimator is as follows:

$$\hat{f}(x) = \frac{1}{nh} \sum_{i=1}^{n} \kappa\left(\frac{x - x_i}{h}\right)$$

Bandwidth

Bandwidth Comparison



Historical (Empirical) Distributions

- The Kernel density estimator is as follows:

$$\hat{f}(x) = \frac{1}{nh}\sum_{i=1}^{n} \kappa\left(\frac{x - x_i}{h}\right)$$

Kernel Function

- Typical Kernel functions:
  1. Normal
  2. Quadratic
  3. Triangular
  4. Epanechnikov

Historical (Empirical) Distributions

- The Kernel density estimator is as follows:

$$\hat{f}(x) = \frac{1}{nh}\sum_{i=1}^{n} \kappa\left(\frac{x - x_i}{h}\right)$$

Kernel Function

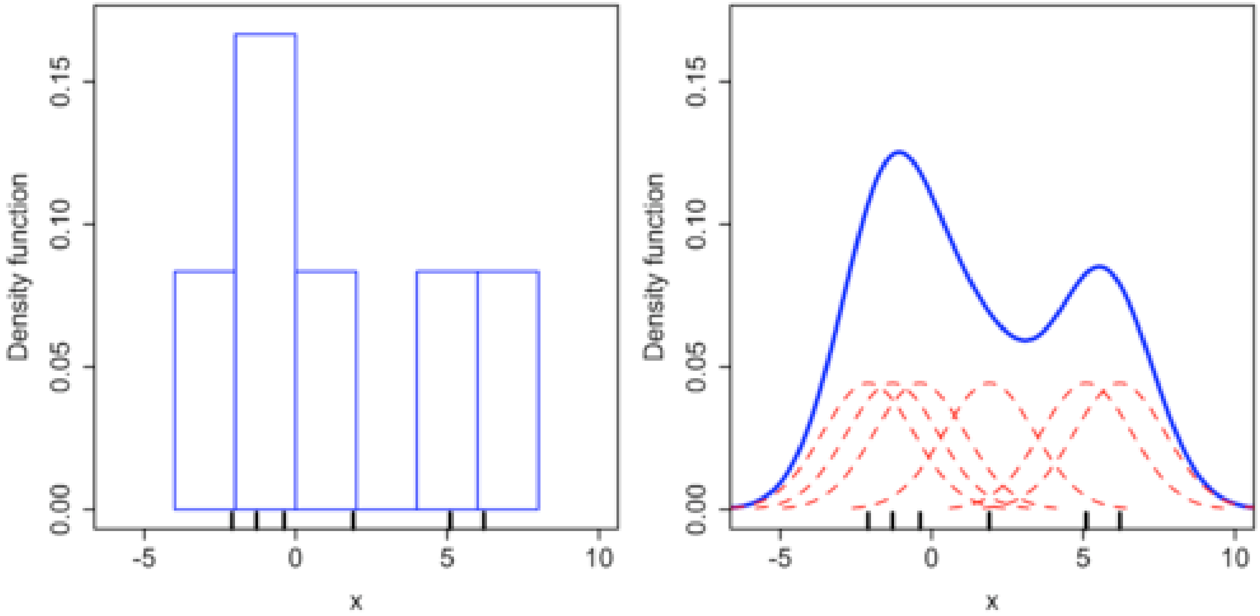
- Typical Kernel functions:
  1. Normal → Common default in a lot of software
  2. Quadratic
  3. Triangular
  4. Epanechnikov

Historical (Empirical) Distributions

- The Kernel density estimator is as follows:

$$\hat{f}(x) = \frac{1}{nh}\sum_{i=1}^{n} \kappa\left(\frac{x - x_i}{h}\right)$$

- Assume Normal Kernel function:



Historical (Empirical) Distributions

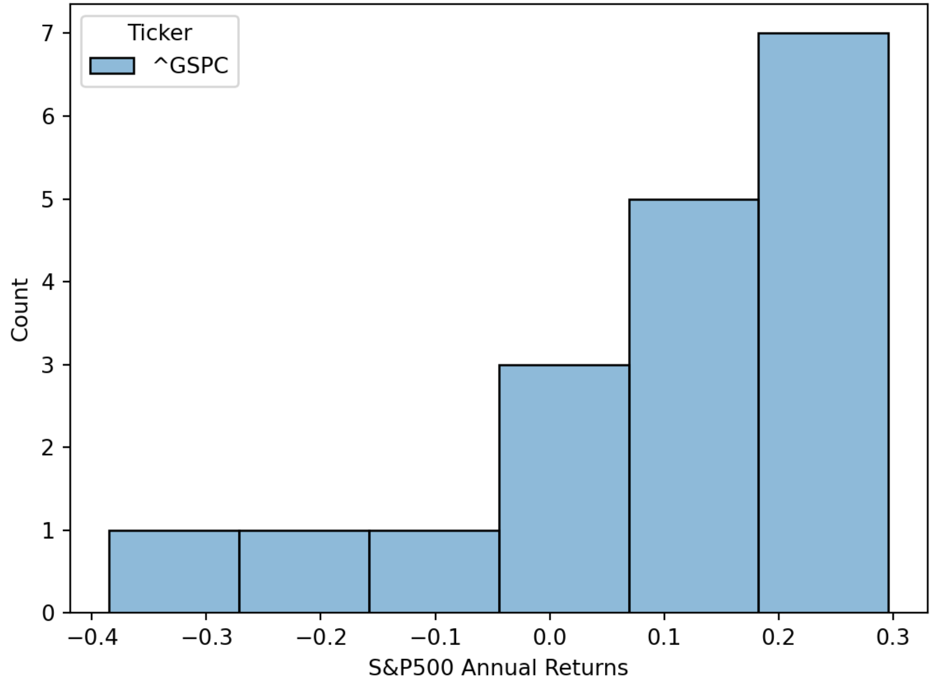
- The Kernel density estimator is as follows:

$$\hat{f}(x) = \frac{1}{nh} \sum_{i=1}^{n} \kappa\left(\frac{x - x_i}{h}\right)$$

- Once you have the Kernel density function, you can sample from this density function.

Historical (Empirical) Distributions – Python

```
import yfinance as yf
GSPC = yf.download("^GSPC", start = "2007-01-01")
gspc_r = GSPC['Close'].resample('YE').ffill().pct_change()
gspc_r = gspc_r.drop(gspc_r.index[0])
```

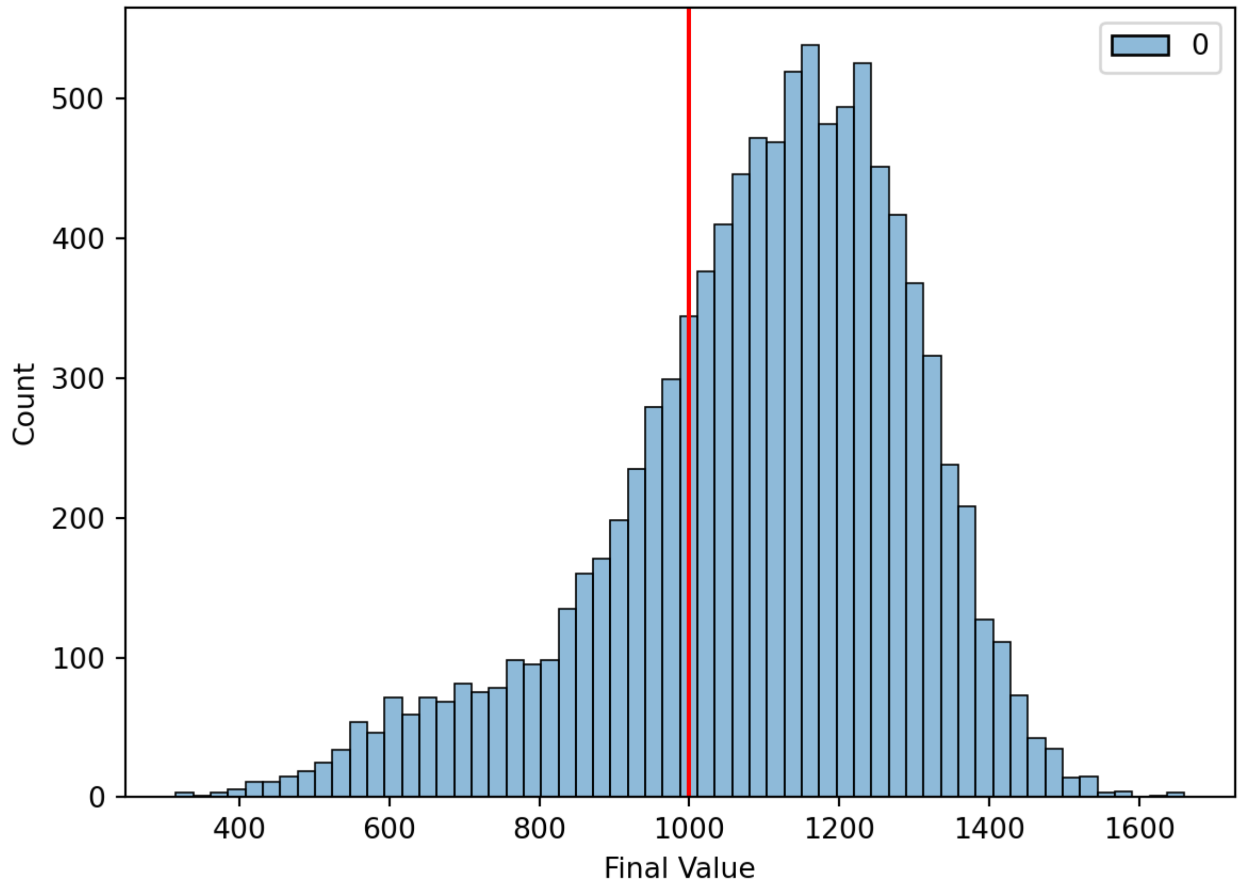


Historical (Empirical) Distributions – Python

```
from scipy.stats import gaussian_kde
kernel = gaussian_kde(gspc_r['^GSPC'])


data = []
for i in range(10000):
  ret = kernel.resample(size = 1).item()
  P0 = 1000
  P1 = P0 * (1 + ret)
  data.append(P1)
df = pd.DataFrame(data)
```

Historical (Empirical) Distributions – Python

```
ax = sns.histplot(data = df)
ax.set(xlabel = "Final Value")
ax.axvline(x = 1000, color = "red")
```



Historical (Empirical) Distributions

- The Kernel density estimator is as follows:

$$\hat{f}(x) = \frac{1}{nh}\sum_{i=1}^{n} \kappa\left(\frac{x - x_i}{h}\right)$$

- Once you have the Kernel density function, you can sample from this density function.

- WARNING: Sample size matters!
  1. If you have large sample sizes, your bandwidth can be smaller and your estimates more accurate.
  2. If you have small sample sizes, your bandwidth increases and estimates are more smoothed.

Hypothesized Future Distribution

- Maybe you know of an upcoming change that will occur in your variable so that the past information is not going to be the future distribution.
- Example:
  - The volatility of the market is forecasted to increase, so instead of a standard deviation of 18% it is 20%.
- In these situations, you can select any distribution of choice.

COMPOUNDING AND CORRELATIONS

Multiple Input Probability Distributions

- Complication arises when you are now simulating multiple inputs changing at the same time.
- Even when the distributions of these inputs are the same, the final result can still be hard to mathematically calculate – benefit of simulation.

Multiple Input Probability Distributions

- General Facts:
  1. When a constant is added to a **random variable** (the variable with the distribution) then the distribution is the same, only shifted by the constant.
  2. The addition of many distributions that are the same is rarely the same shape of distribution – exception would be INDEPENDENT Normal distributions.
  3. The product of many distributions that are the same is rarely the same shape of distribution – exception would be INDEPENDENT lognormal distributions (popular in finance for this reason).

Example

- You want to invest $1,000 in the US stock market for **thirty** years.
- You invest in a mutual fund that tries to produce the same return as the S&P500 Index.

$$P_t = P_0 * (1 + r_{0,1})(1 + r_{1,2})(1 + r_{2,3}) \dots (1 + r_{t-1,t})$$

Example

- You want to invest $1,000 in the US stock market for **thirty** years.
- You invest in a mutual fund that tries to produce the same return as the S&P500 Index.

$$P_t = P_0 * (1 + r_{0,1})(1 + r_{1,2})(1 + r_{2,3}) \ldots (1 + r_{t-1,t})$$

Annual Returns

Example

- You want to invest $1,000 in the US stock market for **thirty** years.
- You invest in a mutual fund that tries to produce the same return as the S&P500 Index.

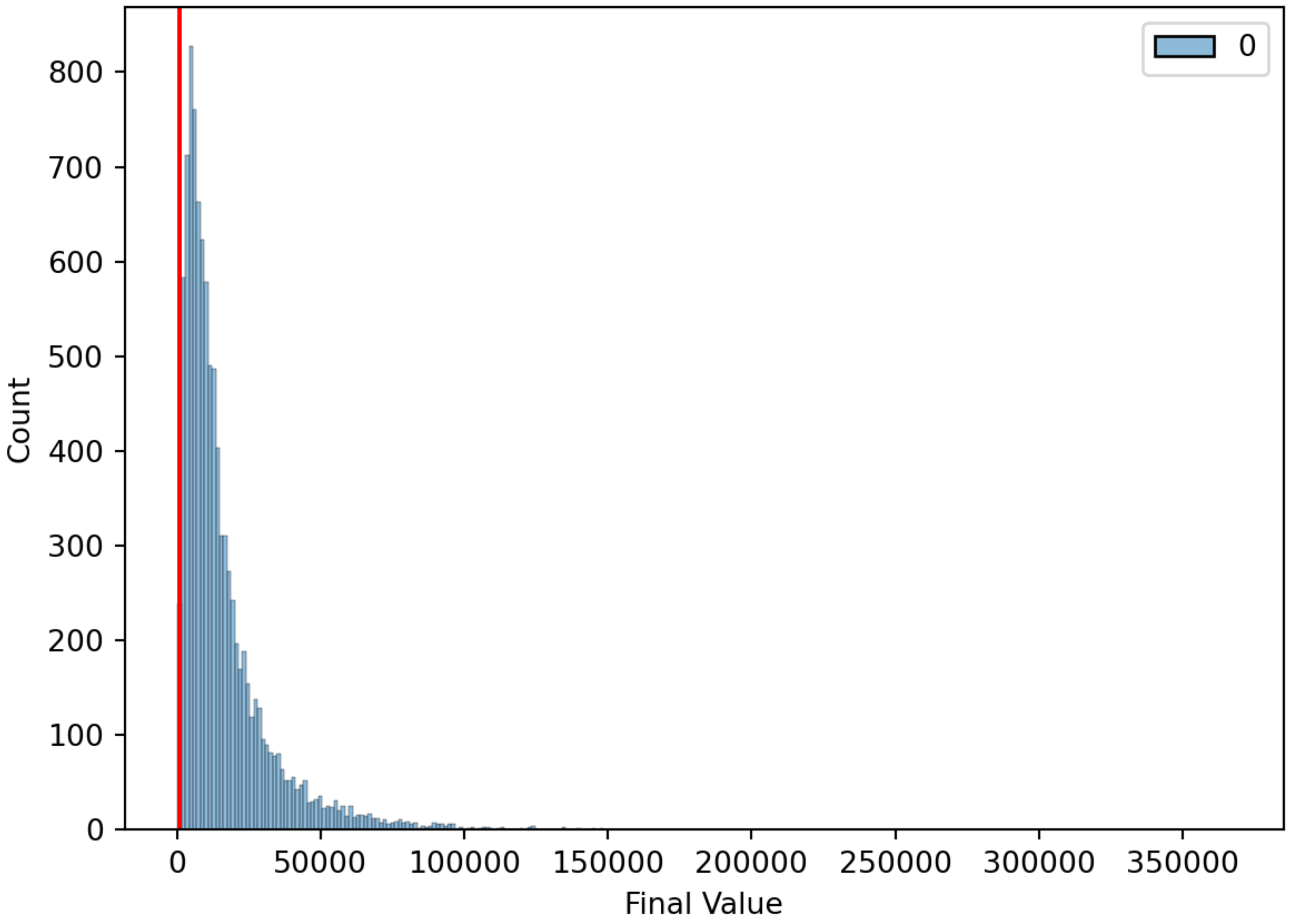
$$P_t = P_0 * (1 + r_{0,1})(1 + r_{1,2})(1 + r_{2,3}) \dots (1 + r_{t-1,t})$$

- Assume annual returns follow a Normal distribution with historical mean of 9.75% and std. dev. of 18% every year.

Multiple Input Prob. Distribution – Python

```
np.random.seed(112358)
data = []
for i in range(10000):
  ret = np.random.normal(loc = 0.0975, scale = 0.18)
  P0 = 1000
  Pt = P0 * (1 + ret)
  for j in range(29):
    ret = np.random.normal(loc = 0.0975, scale = 0.18)
    P0 = 1000
    Pt = Pt * (1 + ret)
  data.append(Pt)
df = pd.DataFrame(data)
```

Example



0
800
700
600
500
400
300
200
100
0
Count
0
50000
100000
150000
200000
250000
300000
350000
Final Value

Correlated Inputs

- Not all inputs are independent of each other.
- Having correlations between your input variables adds even more complication to the simulation and final distribution.
- May need to simulate random variables that have correlation with each other.

Example

- You want to invest $1,000 in the US stock market **or US Treasury bonds** for **thirty** years.
- You invest a certain percentage in a mutual fund that tries to produce the same return as the S&P500 Index and the rest in US Treasury bonds.

$$P_{t,S} = P_{0,S} * (1 + r_{0,1})(1 + r_{1,2})(1 + r_{2,3}) \dots (1 + r_{t-1,t})$$

$$P_{t,B} = P_{0,B} * (1 + r_{0,1})(1 + r_{1,2})(1 + r_{2,3}) \dots (1 + r_{t-1,t})$$

$$P_t = P_{t,S} + P_{t,B}$$

Example

- You want to invest $1,000 in the US stock market **or US Treasury bonds** for **thirty** years.
- You invest a certain percentage in a mutual fund that tries to produce the same return as the S&P500 Index and the rest in US Treasury bonds.
- Treasury bonds perceived as safer investment so when stock market does poorly more people invest in bonds – negatively correlated.

- Assume mutual fund Normal(9.75%, 18%).
- Assume Treasury Bond Normal(4.00%, 7.00%).
- Assume correlation of -0.2.

Adding Correlation

- One way to “add” correlation to data is to multiply the correlation into the data through matrix multiplication (linear algebra!).
- One variable example:
  - X ~ N(mean = 3, var = 2)
  - Want to have a variance of 4
  - What can we do?

Adding Correlation

- One way to "add" correlation to data is to multiply the correlation into the data through matrix multiplication (linear algebra!).
- One variable example:
  - X ~ N(mean = 3, var = 2)
  - Want to have a variance of 4
  - What can we do?
    1. Standardize X → $\frac{X-3}{\sqrt{2}}$ → Z ~ N(mean = 0, var = 1)
    2. Multiply Z by $\sqrt{4}$ → $\sqrt{4}Z$ → Y ~ N(mean = 0, var = 4)
    3. Convert Y back → $Y + 3$ → Y ~ N(mean = 3, var = 4) ← Same mean as X, but now has larger variance!

Adding Correlation

- For multiple variables at the same time, we can use the variance matrix instead:
  - **X** has 2 columns with correlation matrix $\Sigma = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$
  - Want to have a variance matrix of $\Sigma^* = \begin{bmatrix} 1 & -0.2 \\ -0.2 & 1 \end{bmatrix}$
  - What can we do?
    1. Standardize each column of **X** → means = 0, variances = 1 in **Z**
    2. Multiply **Z** by "square root" of Σ^* (Cholesky Decomposition)
    3. Convert **Z** back → means and variances back to what they were before to get **Y**

Cholesky Decomposition

- What is the square root of a number?
  - The square root is a number(s) that when multiplied by itself gives you the original value.
  - Ex: Square root of 4 is either -2 or 2 since both of those numbers when multiplied by themselves equal 4.
- What is the square root of a matrix?
  - The "square root" of a matrix is a matrix that when multiplied by itself gives you the original matrix.
  - This is called a **Cholesky decomposition**.
  - Ex: Cholesky decomp of $\Sigma^* = \begin{bmatrix} 1 & -0.2 \\ -0.2 & 1 \end{bmatrix}$ is

    $L = \begin{bmatrix} 1 & 0 \\ -0.2 & 0.98 \end{bmatrix}$ since $L \times L^T = \begin{bmatrix} 1 & -0.2 \\ -0.2 & 1 \end{bmatrix}$

Cholesky Decomposition

- How does it work in idea?
  - Takes the first column and leaves it alone. “Bends” the second column to be more correlated with the first.
- Cholesky decomposition works best when variables are normally distributed.
- It will be OK if they are symmetric and unimodal.
- If not either, put the column you want unchanged the most first.

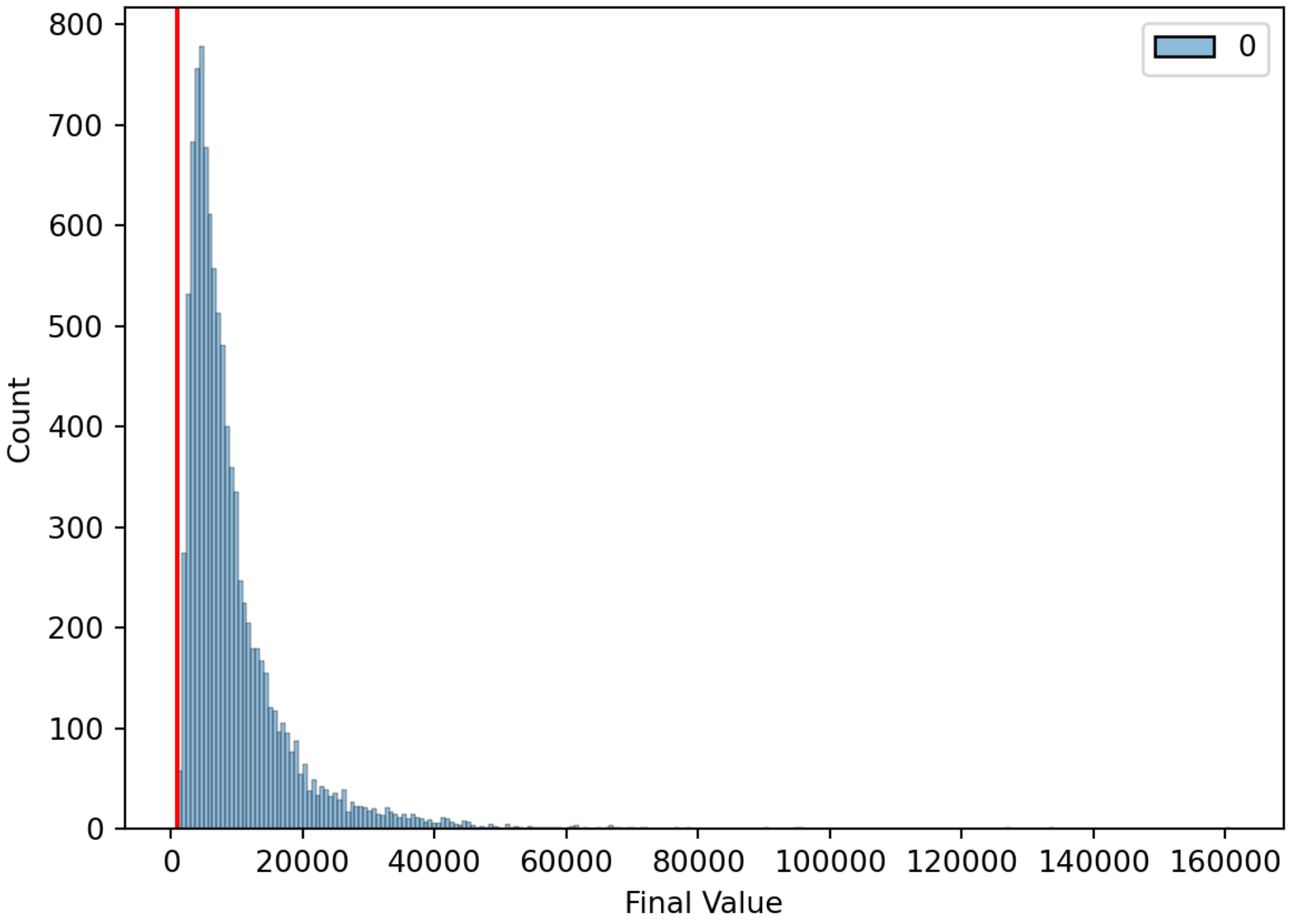
Correlated Inputs – Python

```
import scipy as sp

data = []
R = np.array([[1, -0.2], [-0.2, 1]])
U = sp.linalg.cholesky(R, lower = False)

Perc_B = 0.5
Perc_S = 0.5
Initial = 1000

def standardize(x):
  x_std = (x - np.mean(x))/np.std(x)
  return(x_std)

def destandardize(x_std, x):
  x_old = (x_std * np.std(x)) + np.mean(x)
  return(x_old)
```

Correlated Inputs – Python

```
np.random.seed(112358)

data = []
for i in range(10000):
  S_ret = np.random.normal(loc = 0.0975, scale = 0.18, size = 30)
  B_ret = np.random.normal(loc = 0.04, scale = 0.07, size = 30)
  Both_ret = np.array([standardize(S_ret), standardize(B_ret)])
  SB_ret = U @ Both_ret

  final_SB_ret = np.array([destandardize(SB_ret[0], S_ret), destandardize(SB_ret[1], B_ret)])

  Pt_B = Initial*Perc_B
  Pt_S = Initial*Perc_S

  for j in range(30):
    Pt_S = Pt_S * (1 + final_SB_ret[0][j].item())
    Pt_B = Pt_B * (1 + final_SB_ret[1][j].item())

  Value = Pt_S + Pt_B
  data.append(Value)
```

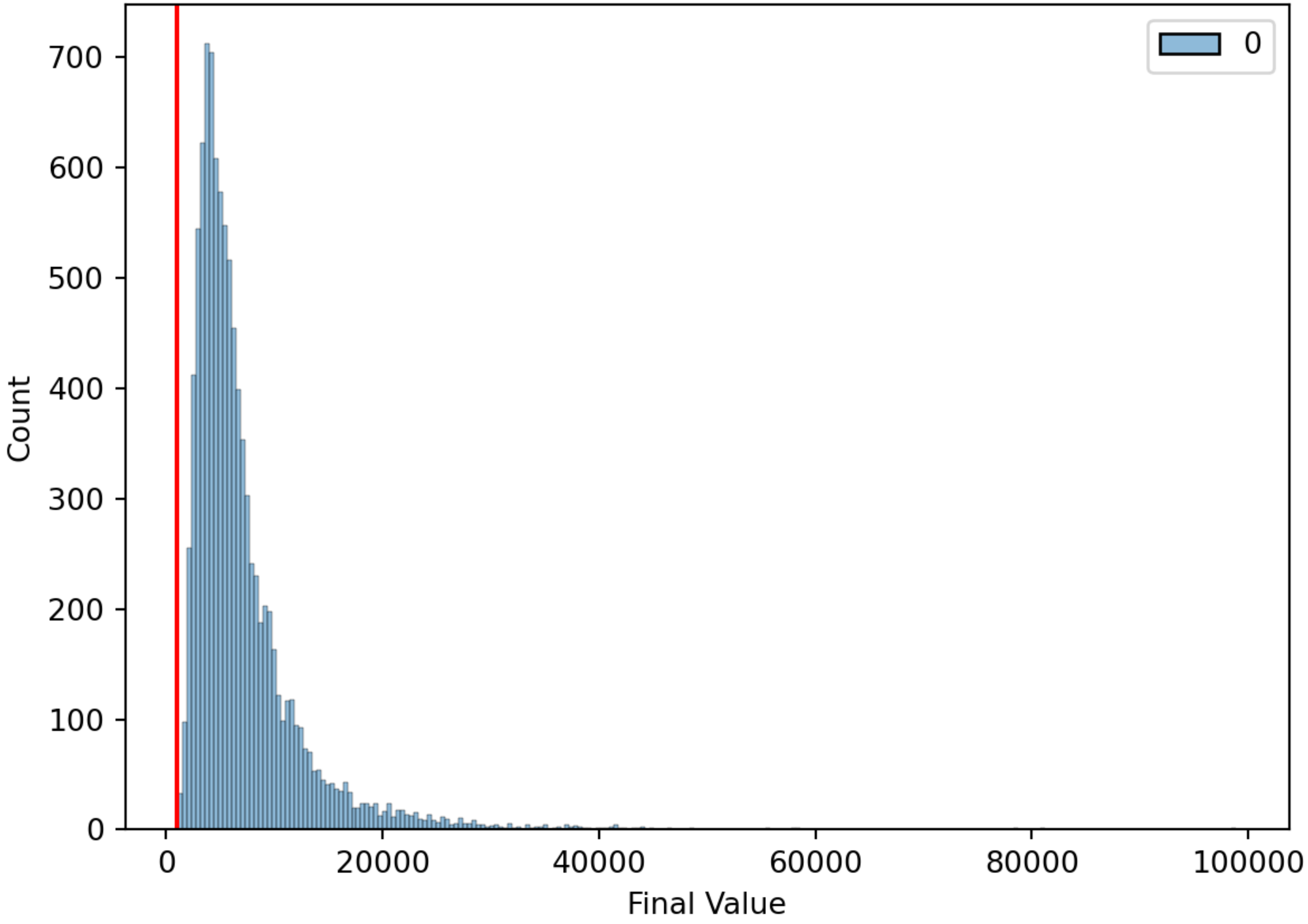
Correlated Inputs – Python



Evaluating Decisions

- Careful about only using summary statistics to evaluate the decisions to be made from simulations.
- Need to look at whole picture – whole distribution.
- Example:
  - Which is “better” – 50/50 stocks/bonds (Strategy A) or 30/70 stocks/bonds (Strategy B)?

Evaluating Decisions – Python



Evaluating Decisions

- Careful about only using summary statistics to evaluate the decisions to be made from simulations.
- Need to look at whole picture – whole distribution.
- Example:
  - Which is “better” – 50/50 stocks/bonds (Strategy A) or 30/70 stocks/bonds (Strategy B)?
  - Mean return of Strategy A – $9,600
  - Mean return of Strategy B – $7,057
  - C.V. of returns for Strategy A – 90.69%
  - C.V. of returns for Strategy B – 74.92%

Evaluating Decisions

- Careful about only using summary statistics to evaluate the decisions to be made from simulations.
- Need to look at whole picture – whole distribution.
- Example:
  - Which is “better” – 50/50 stocks/bonds (Strategy A) or 30/70 stocks/bonds (Strategy B)?
  - Mean return of Strategy A – $9,600
  - Mean return of Strategy B – $7,057
  - C.V. of returns for Strategy A – 90.69%
  - C.V. of returns for Strategy B – 74.92%
  - Strategy A has higher return but APPEARS riskier.

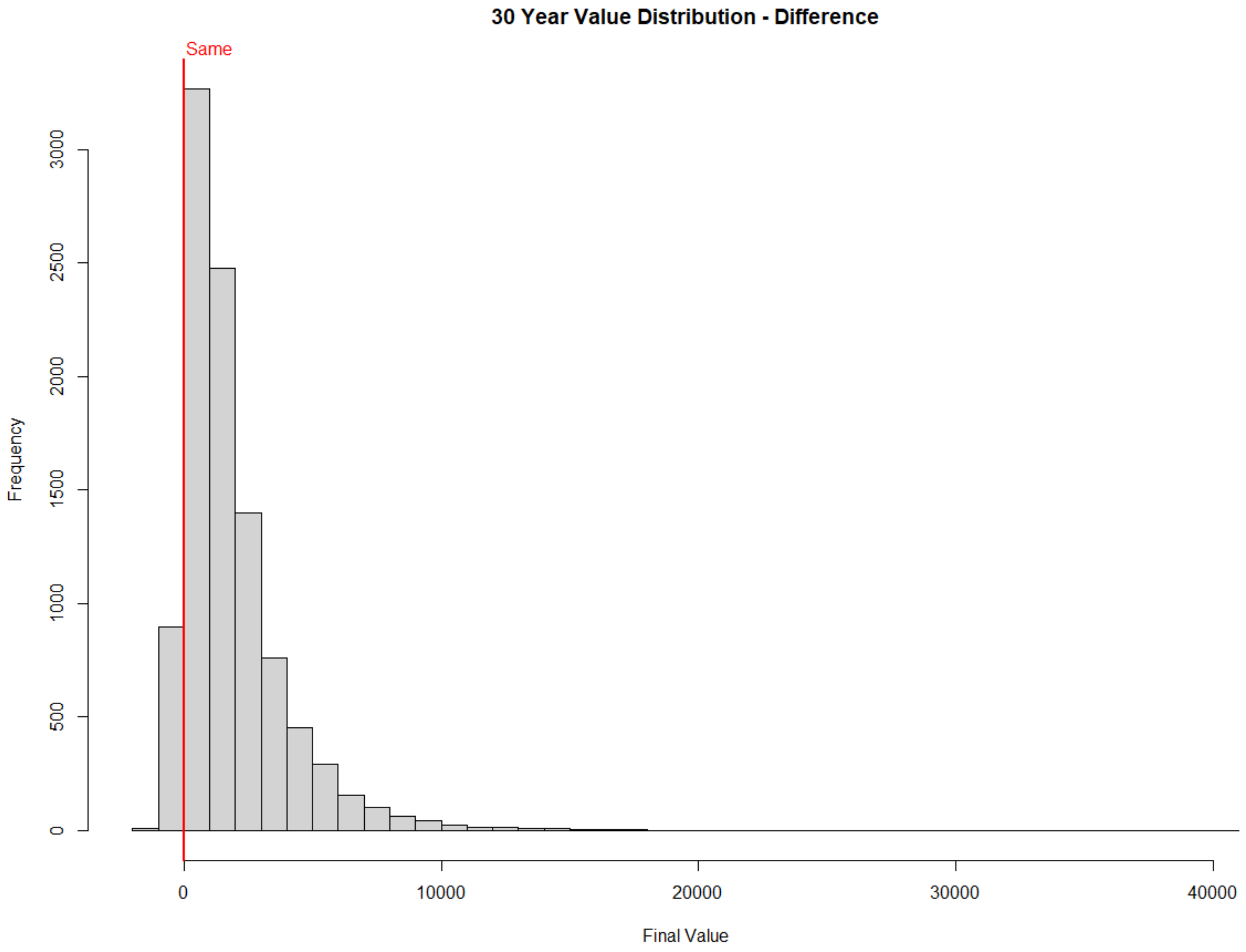
Evaluating Decisions

- Careful about only using summary statistics to evaluate the decisions to be made from simulations.
- Need to look at whole picture – whole distribution.
- Example:
  - Which is “better” – 50/50 stocks/bonds (Strategy A) or 30/70 stocks/bonds (Strategy B)?
  - 5th Percentile of Strategy A – $2,555
  - 5th Percentile of Strategy B – $2,500
  - 95th Percentile of Strategy A – $25,959
  - 95th Percentile of Strategy B – $16,759
  - Strategy A has **less** downside, but **higher** upside.

Evaluating Decisions

- Careful about only using summary statistics to evaluate the decisions to be made from simulations.
- Need to look at whole picture – whole distribution.
- Standard deviation is not always a good measure of riskiness.
- Higher standard deviation not necessarily bad if the largest deviations from the mean are on the upside!

Difference (A – B)



?

HOW MANY AND HOW LONG?

Accuracy vs. Time

- The possible number of outcomes for a simulation output variable is basically infinite.
- We need to get a “sampling” of these values.
- Accuracy of the estimates depends on the number of simulated values.
- **How many simulations do you need to run?**

Accuracy vs. Time

- **How many simulations do you need to run?**
- Confidence interval theory in statistics helps reveal the relationship between accuracy and number of simulations.
- Imagine you are interested in the mean value of the output distribution from your simulation.
- We know the margin of error of the mean!

Accuracy vs. Time

- **How many simulations do you need to run?**
- Confidence interval theory in statistics helps reveal the relationship between accuracy and number of simulations.
- Imagine you are interested in the mean value of the output distribution from your simulation.
- We know the margin of error of the mean!

$$MoE(\bar{x}) = t^* \frac{s}{\sqrt{n}}$$

Accuracy vs. Time

- **How many simulations do you need to run?**
- Confidence interval theory in statistics helps reveal the relationship between accuracy and number of simulations.
- Imagine you are interested in the mean value of the output distribution from your simulation.
- We know the margin of error of the mean!

$$MoE(\bar{x}) = t^* \frac{s}{\sqrt{n}}$$

Number of simulated values

Standard deviation from simulated values

Accuracy vs. Time

- **How many simulations do you need to run?**
- Confidence interval theory in statistics helps reveal the relationship between accuracy and number of simulations.
- Imagine you are interested in the mean value of the output distribution from your simulation.
- We know the margin of error of the mean!

$$MoE(\bar{x}) = t^* \frac{s}{\sqrt{n}}$$

- To **double the accuracy** we need to approximately **quadruple the number of scenarios**.

Accuracy vs. Time

- **How many simulations do you need to run?**
- Confidence interval theory in statistics helps reveal the relationship between accuracy and number of simulations.
- Imagine you are interested in the mean value of the output distribution from your simulation.
- We know the margin of error of the mean!

$$MoE(\bar{x}) = t^* \frac{s}{\sqrt{n}}$$

- To **double the accuracy** we need to approximately **quadruple the number of scenarios**.