

Algorithms for Discrete Random Variables

Outline

Data
Structures and
Simple
Algorithms

Sums of
Independent
Random
Variables

Order
Statistics

1 Data Structures and Simple Algorithms

2 Sums of Independent Random Variables

3 Order Statistics

Data structure for discrete random variables?

- Looks like data structure for continuous RVs:

$$[[f(x)], [support], ["Discrete", "XXX"]]$$

where XXX is PDF, CDF, SF, HF, CHF, or IDF

- Let X denote the number of heads in two tosses of a fair coin; $X \sim \text{Binomial}(n = 2, p = 1/2)$

```
X := [[1/4, 1/2, 1/4], [0, 1, 2],  
      ["Discrete", "PDF"]];
```

```
X := [[x -> binomial(2, 1/2)*(1/2)^2], [0..2],  
      ["Discrete", "PDF"]];
```

Data structure for discrete random variables?

- Looks like data structure for continuous RVs:

```
[[f(x)], [support], ["Discrete", "XXX"]]
```

where XXX is PDF, CDF, SF, HF, CHF, or IDF

- Let X denote the number of heads in two tosses of a fair coin; $X \sim \text{Binomial}(n = 2, p = 1/2)$

```
X := [[1/4, 1/2, 1/4], [0, 1, 2],  
      ["Discrete", "PDF"]];
```

```
X := [[x -> binomial(2, 1/2)*(1/2)^2], [0..2],  
      ["Discrete", "PDF"]];
```

Data structure for discrete random variables?

- Looks like data structure for continuous RVs:

$$[[f(x)], [support], ["Discrete", "XXX"]]$$

where XXX is PDF, CDF, SF, HF, CHF, or IDF

- Let X denote the number of heads in two tosses of a fair coin; $X \sim \text{Binomial}(n = 2, p = 1/2)$

```
X := [[1/4, 1/2, 1/4], [0, 1, 2],  
      ["Discrete", "PDF"]];
```

```
X := [[x -> binomial(2, 1/2)*(1/2)^2], [0..2],  
      ["Discrete", "PDF"]];
```

Data structure for discrete random variables?

- Looks like data structure for continuous RVs:

$$[[f(x)], [support], ["Discrete", "XXX"]]$$

where XXX is PDF, CDF, SF, HF, CHF, or IDF

- Let X denote the number of heads in two tosses of a fair coin; $X \sim \text{Binomial}(n = 2, p = 1/2)$

$$X := [[1/4, 1/2, 1/4], [0, 1, 2],$$
$$["Discrete", "PDF"]];$$
$$X := [[x \rightarrow \text{binomial}(2, 1/2) * (1/2)^2], [0..2],$$
$$["Discrete", "PDF"]];$$

Data structure for discrete random variables?

Two support formats: *NoDot* versus *Dot*

NoDot: Random variables whose **finite supports** display **no pattern**

Example

$$f(x) = \begin{cases} 0.2 & x = 1 \\ 0.5 & x = 7/2 \\ 0.3 & x = 11. \end{cases}$$

RV's probabilities and supports are entered as Maple lists:

```
X := [[0.2, 0.5, 0.3], [1, 7/2, 11],  
      ["Discrete", "PDF"]];
```

Data structure for discrete random variables?

Two support formats: *NoDot* versus *Dot*

NoDot: Random variables whose **finite supports** display **no pattern**

Example

$$f(x) = \begin{cases} 0.2 & x = 1 \\ 0.5 & x = 7/2 \\ 0.3 & x = 11. \end{cases}$$

RV's probabilities and supports are entered as Maple lists:

```
X := [[0.2, 0.5, 0.3], [1, 7/2, 11],  
      ["Discrete", "PDF"]];
```

Data structure for discrete random variables?

Two support formats: *NoDot* versus *Dot*

NoDot: Random variables whose **finite supports** display **no pattern**

Example

$$f(x) = \begin{cases} 0.2 & x = 1 \\ 0.5 & x = 7/2 \\ 0.3 & x = 11. \end{cases}$$

RV's probabilities and supports are entered as Maple lists:

```
X := [[0.2, 0.5, 0.3], [1, 7/2, 11],  
      ["Discrete", "PDF"]];
```

Data structure for discrete random variables?

Dot: Random variables with **patterned** supports

Example

$$f(x) = \binom{5}{x} (0.2)^x (0.8)^{5-x} \quad x = 0, 1, \dots, 5.$$

If RV's support is Ω , the general *Dot* format is:

$$[\min\{\Omega\} .. \max\{\Omega\}, k, x \rightarrow g(x)]$$

with defaults $k = 1$ and $g(x) = x$.

```
X := [[x -> binomial(5, x) * (0.8)^5 * 0.2^(5-x)],  
      [0..5], ["Discrete", "PDF"]];
```

Data structure for discrete random variables?

Dot: Random variables with **patterned** supports

Example

$$f(x) = \binom{5}{x} (0.2)^x (0.8)^{5-x} \quad x = 0, 1, \dots, 5.$$

If RV's support is Ω , the general *Dot* format is:

$$[\min\{\Omega\} .. \max\{\Omega\}, k, x \rightarrow g(x)]$$

with defaults $k = 1$ and $g(x) = x$.

```
X := [[x -> binomial(5, x) * (0.8)^5 * 0.2^(5-x)],  
      [0..5], ["Discrete", "PDF"]];
```

Data structure for discrete random variables?

Dot: Random variables with **patterned** supports

Example

$$f(x) = \binom{5}{x} (0.2)^x (0.8)^{5-x} \quad x = 0, 1, \dots, 5.$$

If RV's support is Ω , the general *Dot* format is:

$$[\min\{\Omega\} .. \max\{\Omega\}, k, x \rightarrow g(x)]$$

with defaults $k = 1$ and $g(x) = x$.

```
X := [[x -> binomial(5, x) * (0.8)^5 * 0.2^(5-x)],  
      [0..5], ["Discrete", "PDF"]];
```

Simple Algorithms

Algorithms for
Discrete
Random
Variables

Outline

Data
Structures and
Simple
Algorithms

Sums of
Independent
Random
Variables

Order
Statistics

Converting functional forms

	$f(x_i)$	$F(x_i)$	$S(x_i)$	$h(x_i)$	$H(x_i)$
$f(x)$	•	$\sum_{j x_j \leq x_i} f(x_j)$	$\sum_{j x_j \geq x_i} f(x_j)$	$\frac{f(x_i)}{\sum_{j x_j \geq x_i} f(x_j)}$	$-\log \left(\sum_{j x_j \geq x_i} f(x_j) \right)$
$F(x)$	$F(x_i) - F(x_{i-1})$	•	$1 - F(x_{i-1})$	$\frac{F(x_i) - F(x_{i-1})}{1 - F(x_{i-1})}$	$-\log(1 - F(x_{i-1}))$
$S(x)$	$S(x_i) - S(x_{i+1})$	$1 - S(x_{i+1})$	•	$\frac{S(x_i) - S(x_{i+1})}{S(x_i)}$	$-\log S(x_i)$
$h(x)$	$h(x_i) \prod_{j x_j < x_i} [1 - h(x_j)]$	$1 - \prod_{j x_j \leq x_i} [1 - h(x_j)]$	$\prod_{j x_j < x_i} [1 - h(x_j)]$	•	$-\sum_{j x_j < x_i} \log(1 - h(x_j))$
$H(x)$	$e^{-H(x_i)} - e^{-H(x_{i+1})}$	$1 - e^{-H(x_{i+1})}$	$e^{-H(x_i)}$	$\frac{e^{-H(x_i)} - e^{-H(x_{i+1})}}{e^{-H(x_i)}}$	•

Simple Algorithms

Algorithms for
Discrete
Random
Variables

Outline

Data
Structures and
Simple
Algorithms

Sums of
Independent
Random
Variables

Order
Statistics

Example

Given the hazard function

$$h(x) = 1/4 \quad x = 1, 2, \dots,$$

find $f(x_i)$ for $x_i = 1, 2, 3, \dots$.

Using the previous table,

$$f(x_i) = h(x_i) \prod_{j|x_j < x_i} [1 - h(x_j)] = \frac{1}{4} \prod_{j=1}^{x_i-1} \left[1 - \frac{1}{4}\right] = \frac{1}{4} \left(\frac{3}{4}\right)^{x_i-1}$$

for $x_i = 1, 2, 3, \dots$, which is a geometric distribution with $p = 1/4$.

Simple Algorithms

Algorithms for
Discrete
Random
Variables

Outline

Data
Structures and
Simple
Algorithms

Sums of
Independent
Random
Variables

Order
Statistics

Example

Given the hazard function

$$h(x) = 1/4 \quad x = 1, 2, \dots,$$

find $f(x_i)$ for $x_i = 1, 2, 3, \dots$.

Using the previous table,

$$f(x_i) = h(x_i) \prod_{j|x_j < x_i} [1 - h(x_j)] = \frac{1}{4} \prod_{j=1}^{x_i-1} \left[1 - \frac{1}{4}\right] = \frac{1}{4} \left(\frac{3}{4}\right)^{x_i-1}$$

for $x_i = 1, 2, 3, \dots$, which is a geometric distribution with $p = 1/4$.

Simple Algorithms

Algorithms for
Discrete
Random
Variables

Outline

Data
Structures and
Simple
Algorithms

Sums of
Independent
Random
Variables

Order
Statistics

Example

Let

$$f(x) = (5 - x)/10 \quad x = 1, 2, 3, 4$$

be the PDF of the discrete random variable X . Use `PlotDist` to depict the CDF of X .

```
X := [[x -> (5 - x) / 10], [1 .. 4],  
      ["Discrete", "PDF"]];  
  
PlotDist(CDF(X));
```

Simple Algorithms

Algorithms for
Discrete
Random
Variables

Outline

Data
Structures and
Simple
Algorithms

Sums of
Independent
Random
Variables

Order
Statistics

Example

Let

$$f(x) = (5 - x)/10 \quad x = 1, 2, 3, 4$$

be the PDF of the discrete random variable X . Use `PlotDist` to depict the CDF of X .

```
X := [[x -> (5 - x) / 10], [1 .. 4],  
      ["Discrete", "PDF"]];
```

```
PlotDist(CDF(X));
```

Simple Algorithms

Algorithms for
Discrete
Random
Variables

Outline

Data
Structures and
Simple
Algorithms

Sums of
Independent
Random
Variables

Order
Statistics

Example

Let

$$f(x) = (5 - x)/10 \quad x = 1, 2, 3, 4$$

be the PDF of the discrete random variable X . Use `PlotDist` to depict the CDF of X .

```
X := [[x -> (5 - x) / 10], [1 .. 4],  
      ["Discrete", "PDF"]];
```

```
PlotDist(CDF(X));
```

Simple Algorithms

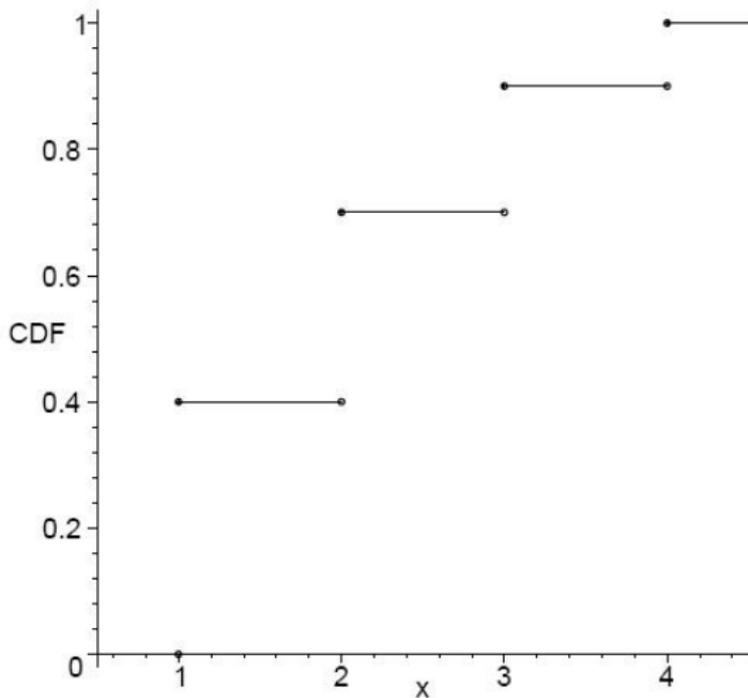
Algorithms for
Discrete
Random
Variables

Outline

Data
Structures and
Simple
Algorithms

Sums of
Independent
Random
Variables

Order
Statistics



CDF for $f(x) = (5 - x)/10$ for $x = 1, 2, 3, 4$.

Simple Algorithms

Algorithms for
Discrete
Random
Variables

Outline

Data
Structures and
Simple
Algorithms

Sums of
Independent
Random
Variables

Order
Statistics

Others simple algorithmic procedures include:

- A procedure that verifies the validity of a PDF; i.e., it checks

$$\sum_{\text{all } x_j \in \Omega} \Pr(X = x_j) = 1, \quad \Pr(X = x_j) > 0 \text{ for each } x_j \in \Omega$$

- Procedures for computing the mean, variance, skewness, and kurtosis of a random variable using the `ExpectedValue(X)` procedure
- Extending the `ExpectedValue(X)` procedure to compute the expected value of a function of a random variable

Simple Algorithms

Algorithms for
Discrete
Random
Variables

Outline

Data
Structures and
Simple
Algorithms

Sums of
Independent
Random
Variables

Order
Statistics

Example

Let X be a binomial random variable with parameters n and p .
Show that

$$E \left[\frac{1}{X+1} \right] = \frac{1 - (1-p)^{n+1}}{(n+1)p}.$$

```
X := BinomialRV(n, p);
```

```
ExpectedValue(X, x -> 1 / (x + 1));
```

Simple Algorithms

Algorithms for
Discrete
Random
Variables

Outline

Data
Structures and
Simple
Algorithms

Sums of
Independent
Random
Variables

Order
Statistics

Example

Let X be a binomial random variable with parameters n and p .
Show that

$$E \left[\frac{1}{X+1} \right] = \frac{1 - (1-p)^{n+1}}{(n+1)p}.$$

```
X := BinomialRV(n, p);
```

```
ExpectedValue(X, x -> 1 / (x + 1));
```

Simple Algorithms

Algorithms for
Discrete
Random
Variables

Outline

Data
Structures and
Simple
Algorithms

Sums of
Independent
Random
Variables

Order
Statistics

Example

Let X be a binomial random variable with parameters n and p .
Show that

$$E \left[\frac{1}{X+1} \right] = \frac{1 - (1-p)^{n+1}}{(n+1)p}.$$

```
X := BinomialRV(n, p);
```

```
ExpectedValue(X, x -> 1 / (x + 1));
```

Sums of Independent *Discrete* Random Variables

Sums of Independent Discrete Random Variables

- An important operation in probability theory and statistical inference is calculating the distribution of sums of independent random variables
- Most texts devote the majority of their attention to sums of continuous random variables
- This section provides the insight into the part of the Convolution(X , Y) procedure that computes the sum of independent discrete random variables with integer valued supports that are bounded below

Discrete Convolution Formula

There are several approaches for computing the PDF of $Z = X + Y$ in the discrete case.

For example, the event $\{X + Y = z\}$ for integer z can be written as the union of the disjoint events $\{X = \zeta, Y = z - \zeta\}$, $\{X = \zeta + 1, Y = z - (\zeta + 1)\}$, \dots , $\{X = z - \zeta, Y = \zeta\}$, where ζ is the minimum of the union of the support values of X and Y .

$$\begin{aligned}\Pr(Z = z) &= \Pr(X + Y = z) \\ &= \sum_{k=\zeta}^{z-\zeta} \Pr(X = k, Y = z - k) \\ &= \sum_{k=\zeta}^{z-\zeta} \Pr(X = k)\Pr(Y = z - k).\end{aligned}$$

Discrete Convolution Formula

There are several approaches for computing the PDF of $Z = X + Y$ in the discrete case.

For example, the event $\{X + Y = z\}$ for integer z can be written as the union of the disjoint events $\{X = \zeta, Y = z - \zeta\}$, $\{X = \zeta + 1, Y = z - (\zeta + 1)\}$, \dots , $\{X = z - \zeta, Y = \zeta\}$, where ζ is the minimum of the union of the support values of X and Y .

$$\begin{aligned}\Pr(Z = z) &= \Pr(X + Y = z) \\ &= \sum_{k=\zeta}^{z-\zeta} \Pr(X = k, Y = z - k) \\ &= \sum_{k=\zeta}^{z-\zeta} \Pr(X = k)\Pr(Y = z - k).\end{aligned}$$

Discrete Convolution Formula: Practical Example

Algorithms for
Discrete
Random
Variables

Outline

Data
Structures and
Simple
Algorithms

Sums of
Independent
Random
Variables

Order
Statistics

Example

If X and Y are independent Poisson random variables with respective parameters λ_1 and λ_2 , compute the PDF of $Z = X + Y$.

$$\begin{aligned}\Pr(Z = z) &= \sum_{k=0}^z \frac{e^{-\lambda_1} \lambda_1^k}{k!} \cdot \frac{e^{-\lambda_2} \lambda_2^{z-k}}{(z-k)!} \\ &= e^{-(\lambda_1 + \lambda_2)} \sum_{k=0}^z \frac{\lambda_1^k \lambda_2^{z-k}}{k! (z-k)!} \\ &= \frac{e^{-(\lambda_1 + \lambda_2)}}{z!} \sum_{k=0}^z \frac{z!}{k! (z-k)!} \lambda_1^k \lambda_2^{z-k} \\ &= \frac{(\lambda_1 + \lambda_2)^z e^{-(\lambda_1 + \lambda_2)}}{z!} \quad z = 0, 1, 2, \dots\end{aligned}$$

Discrete Convolution Formula: Practical Example

Algorithms for
Discrete
Random
Variables

Outline

Data
Structures and
Simple
Algorithms

Sums of
Independent
Random
Variables

Order
Statistics

Example

If X and Y are independent Poisson random variables with respective parameters λ_1 and λ_2 , compute the PDF of $Z = X + Y$.

$$\begin{aligned}\Pr(Z = z) &= \sum_{k=0}^z \frac{e^{-\lambda_1} \lambda_1^k}{k!} \cdot \frac{e^{-\lambda_2} \lambda_2^{z-k}}{(z-k)!} \\ &= e^{-(\lambda_1 + \lambda_2)} \sum_{k=0}^z \frac{\lambda_1^k \lambda_2^{z-k}}{k! (z-k)!} \\ &= \frac{e^{-(\lambda_1 + \lambda_2)}}{z!} \sum_{k=0}^z \frac{z!}{k! (z-k)!} \lambda_1^k \lambda_2^{z-k} \\ &= \frac{(\lambda_1 + \lambda_2)^z e^{-(\lambda_1 + \lambda_2)}}{z!} \quad z = 0, 1, 2, \dots\end{aligned}$$

Discrete Convolution Formula: Impractical Example

Algorithms for
Discrete
Random
Variables

Outline

Data
Structures and
Simple
Algorithms

Sums of
Independent
Random
Variables

Order
Statistics

Example

Suppose X and Y are independent RVs with PDFs:

$$f_X(x) = \begin{cases} 0.15 & x = -3 \\ 0.25 & x = -1 \\ 0.1 & x = 2 \\ 0.3 & x = 6 \\ 0.2 & x = 8 \end{cases} \quad f_Y(y) = \begin{cases} 0.2 & y = -2 \\ 0.1 & y = 1 \\ 0.3 & y = 5 \\ 0.4 & y = 8. \end{cases}$$

Compute the PDF of $Z = X + Y$.

Not bad enough? Consider the supports of X and Y to be $\{-1002, -15, 2, 62, 211\}$ and $\{-216, -57, 23, 81\}$!

Discrete Convolution Formula: Impractical Example

Algorithms for
Discrete
Random
Variables

Outline

Data
Structures and
Simple
Algorithms

Sums of
Independent
Random
Variables

Order
Statistics

Example

Suppose X and Y are independent RVs with PDFs:

$$f_X(x) = \begin{cases} 0.15 & x = -3 \\ 0.25 & x = -1 \\ 0.1 & x = 2 \\ 0.3 & x = 6 \\ 0.2 & x = 8 \end{cases} \quad f_Y(y) = \begin{cases} 0.2 & y = -2 \\ 0.1 & y = 1 \\ 0.3 & y = 5 \\ 0.4 & y = 8. \end{cases}$$

Compute the PDF of $Z = X + Y$.

Not bad enough? Consider the supports of X and Y to be $\{-1002, -15, 2, 62, 211\}$ and $\{-216, -57, 23, 81\}$!

Discrete Convolution Formula: Impractical Example

To compute $\Pr(Z = 4)$, we can use the discrete convolution formula with $\zeta = -3$

$$\begin{aligned}\Pr(Z = 4) &= \sum_{k=-3}^7 \Pr(X = k, Y = 4 - k) \\ &= \Pr(X = -3) \cdot \Pr(Y = 7) + \Pr(X = -2) \cdot \Pr(Y = 6) + \\ &\quad \Pr(X = -1) \cdot \Pr(Y = 5) + \Pr(X = 0) \cdot \Pr(Y = 4) + \\ &\quad \Pr(X = 1) \cdot \Pr(Y = 3) + \Pr(X = 2) \cdot \Pr(Y = 2) + \\ &\quad \Pr(X = 3) \cdot \Pr(Y = 1) + \Pr(X = 4) \cdot \Pr(Y = 0) + \\ &\quad \Pr(X = 5) \cdot \Pr(Y = -1) + \Pr(X = 6) \cdot \Pr(Y = -2) + \\ &\quad \Pr(X = 7) \cdot \Pr(Y = -3) \\ &= 0.15 \cdot 0 + 0 \cdot 0 + 0.25 \cdot 0.3 + 0 \cdot 0 + 0 \cdot 0 + 0.1 \cdot 0 + \\ &\quad 0 \cdot 0.1 + 0 \cdot 0 + 0 \cdot 0 + 0.3 \cdot 0.2 + 0 \cdot 0 \\ &= 0 + 0 + 0.075 + 0 + 0 + 0 + 0 + 0 + 0 + 0.06 + 0 \\ &= 0.135.\end{aligned}$$

Discrete Convolution Formula: Impractical Example

To compute $\Pr(Z = 4)$, we can use the discrete convolution formula with $\zeta = -3$

$$\begin{aligned}\Pr(Z = 4) &= \sum_{k=-3}^7 \Pr(X = k, Y = 4 - k) \\ &= \Pr(X = -3) \cdot \Pr(Y = 7) + \Pr(X = -2) \cdot \Pr(Y = 6) + \\ &\quad \Pr(X = -1) \cdot \Pr(Y = 5) + \Pr(X = 0) \cdot \Pr(Y = 4) + \\ &\quad \Pr(X = 1) \cdot \Pr(Y = 3) + \Pr(X = 2) \cdot \Pr(Y = 2) + \\ &\quad \Pr(X = 3) \cdot \Pr(Y = 1) + \Pr(X = 4) \cdot \Pr(Y = 0) + \\ &\quad \Pr(X = 5) \cdot \Pr(Y = -1) + \Pr(X = 6) \cdot \Pr(Y = -2) + \\ &\quad \Pr(X = 7) \cdot \Pr(Y = -3) \\ &= 0.15 \cdot 0 + 0 \cdot 0 + 0.25 \cdot 0.3 + 0 \cdot 0 + 0 \cdot 0 + 0.1 \cdot 0 + \\ &\quad 0 \cdot 0.1 + 0 \cdot 0 + 0 \cdot 0 + 0.3 \cdot 0.2 + 0 \cdot 0 \\ &= 0 + 0 + 0.075 + 0 + 0 + 0 + 0 + 0 + 0 + 0.06 + 0 \\ &= 0.135.\end{aligned}$$

Moment Generating Function (MGF) Technique

Algorithms for
Discrete
Random
Variables

Outline

Data
Structures and
Simple
Algorithms

Sums of
Independent
Random
Variables

Order
Statistics

If $Z = X + Y$ and X and Y are independent, then MGF of Z is

$$\begin{aligned}M_Z(t) &= E\left(e^{t(X+Y)}\right) \\&= E\left(e^{tX} e^{tY}\right) \\&= E\left(e^{tX}\right) E\left(e^{tY}\right) \\&= M_X(t) M_Y(t).\end{aligned}$$

For the previous example, the MGFs of X and Y are

$$M_X(t) = E\left(e^{tX}\right) = 0.15e^{-3t} + 0.25e^{-t} + 0.1e^{2t} + 0.3e^{6t} + 0.2e^{8t}$$

$$M_Y(t) = E\left(e^{tY}\right) = 0.2e^{-2t} + 0.1e^t + 0.3e^{5t} + 0.4e^{8t}$$

Moment Generating Function (MGF) Technique

Algorithms for
Discrete
Random
Variables

Outline

Data
Structures and
Simple
Algorithms

Sums of
Independent
Random
Variables

Order
Statistics

Result of the product of $M_X(t)$ and $M_Y(t)$:

$$M_Z(t) = 0.03e^{-5t} + 0.05e^{-3t} + 0.015e^{-2t} + 0.045 + 0.045e^{2t} + \\ 0.01e^{3t} + 0.135e^{4t} + 0.06e^{5t} + 0.04e^{6t} + 0.16e^{7t} + 0.02e^{9t} + \\ 0.04e^{10t} + 0.09e^{11t} + 0.06e^{13t} + 0.12e^{14t} + 0.08e^{16t}$$

Considerations:

- Structure of discrete random variables in APPL: Supports are ordered
- Implementation issues with MGFs or PGFs when the supports of the random variables X and/or Y are not integer-valued
- Characteristic functions

Conceptual Algorithm Development

*Algorithms for
Discrete
Random
Variables*

Outline

Data
Structures and
Simple
Algorithms

Sums of
Independent
Random
Variables

Order
Statistics

Ideas:

- If the discrete convolution formula is not used, then some type of sorting will be necessary
- Build arrays to hold sums and corresponding probability values, methodically compute sums and dump the sums in the arrays, sort the arrays

Conceptual Algorithm Development

*Algorithms for
Discrete
Random
Variables*

Outline

Data
Structures and
Simple
Algorithms

Sums of
Independent
Random
Variables

Order
Statistics

Ideas:

- If the discrete convolution formula is not used, then some type of sorting will be necessary
- Build arrays to hold sums and corresponding probability values, methodically compute sums and dump the sums in the arrays, sort the arrays

Conceptual Algorithm Development

Brute Force Method

- Suppose X has support x_1, x_2, \dots, x_n and Y has support y_1, y_2, \dots, y_m .
- Compute all possible sums between the support of X and the support of Y by brute force: $x_1 + y_1, x_1 + y_2, \dots, x_1 + y_m, x_2 + y_1, x_2 + y_2, \dots, x_n + y_{m-1}, x_n + y_m$. Place the sums in an array.
- Sort the array with the sums. When $n \cdot m$ is “small” (less than 20), use basic insertion sort. When $n \cdot m$ is larger, but still less than 100, use more efficient heapsort.

Conceptual Algorithm Development

Algorithms for
Discrete
Random
Variables

Outline

Data
Structures and
Simple
Algorithms

Sums of
Independent
Random
Variables

Order
Statistics

Brute Force Method

- Suppose X has support x_1, x_2, \dots, x_n and Y has support y_1, y_2, \dots, y_m .
- Compute all possible sums between the support of X and the support of Y by brute force: $x_1 + y_1, x_1 + y_2, \dots, x_1 + y_m, x_2 + y_1, x_2 + y_2, \dots, x_n + y_{m-1}, x_n + y_m$. Place the sums in an array.
- Sort the array with the sums. When $n \cdot m$ is “small” (less than 20), use basic insertion sort. When $n \cdot m$ is larger, but still less than 100, use more efficient heapsort.

Conceptual Algorithm Development

Algorithms for
Discrete
Random
Variables

Outline

Data
Structures and
Simple
Algorithms

Sums of
Independent
Random
Variables

Order
Statistics

Brute Force Method

- Suppose X has support x_1, x_2, \dots, x_n and Y has support y_1, y_2, \dots, y_m .
- Compute all possible sums between the support of X and the support of Y by brute force: $x_1 + y_1, x_1 + y_2, \dots, x_1 + y_m, x_2 + y_1, x_2 + y_2, \dots, x_n + y_{m-1}, x_n + y_m$. Place the sums in an array.
- Sort the array with the sums. When $n \cdot m$ is “small” (less than 20), use basic insertion sort. When $n \cdot m$ is larger, but still less than 100, use more efficient heapsort.

Conceptual Algorithm Development

Algorithms for
Discrete
Random
Variables

Outline

Data
Structures and
Simple
Algorithms

Sums of
Independent
Random
Variables

Order
Statistics

For $n \cdot m > 100$, the brute force method is impractical

Idea:

- Instead of constructing the sums array s first and then sorting it, construct s by sequentially appending the next ordered element.
- **“Moving heap method:”** Build, delete, and insert sums into a minimum heap data structure as the sums are computed.

Moving Heap Method

Moving Heap Method

- The idea behind this sorting algorithm is the construction of a two-dimensional “conceptual” array A .
- The array A serves as a bookkeeping device to help explain the nature of the algorithm.
- A is displayed to resemble the axes in the Cartesian coordinate system. We assume that the supports of X and Y are arranged in increasing order; i.e., $x_1 < x_2 < \dots < x_n$ and $y_1 < y_2 < \dots < y_m$. The array cell (i, j) contains the sum $A_{i,j} = y_i + x_j$ for $i = 1, 2, \dots, m$ and $j = 1, 2, \dots, n$.

Moving Heap Method

Algorithms for
Discrete
Random
Variables

Outline

Data
Structures and
Simple
Algorithms

Sums of
Independent
Random
Variables

Order
Statistics

y_m	$y_m + x_1$	$y_m + x_2$	\dots	$y_m + x_n$
	\vdots	\vdots	\ddots	\vdots
y_2	$y_2 + x_1$	$y_2 + x_2$	\dots	$y_2 + x_n$
y_1	$y_1 + x_1$	$y_1 + x_2$	\dots	$y_1 + x_n$
	x_1	x_2		x_n

Array \hat{A} where $x_1 < x_2 < \dots < x_n$ and $y_1 < y_2 < \dots < y_m$.

Moving Heap Method

Algorithms for
Discrete
Random
Variables

Outline

Data
Structures and
Simple
Algorithms

Sums of
Independent
Random
Variables

Order
Statistics

Example

Suppose X and Y are independent RVs with PDFs:

$$f_X(x) = \begin{cases} 0.15 & x = -3 \\ 0.25 & x = -1 \\ 0.1 & x = 2 \\ 0.3 & x = 6 \\ 0.2 & x = 8 \end{cases} \quad f_Y(y) = \begin{cases} 0.2 & y = -2 \\ 0.1 & y = 1 \\ 0.3 & y = 5 \\ 0.4 & y = 8. \end{cases}$$

Use the “moving heap method” to determine the PDF of $Z = X + Y$.

Moving Heap Method

Algorithms for
Discrete
Random
Variables

Outline

Data
Structures and
Simple
Algorithms

Sums of
Independent
Random
Variables

Order
Statistics

- Construct the 5×6 array A . Set $A_{i,6} = 0$ for $i = 1, 2, 3, 4$ and $A_{5,j} = 0$ for $j = 1, 2, 3, 4, 5$.
- The smallest value in A is positioned in cell $(1, 1)$.
- The algorithm designates the cell $(1, 1)$ as an “active” cell.

row 5		1	0	0	0	0	
row 4	8					0	
row 3	5					0	
row 2	1					0	
row 1	-2	-5				1	
		-3	-1	2	6	8	
		col 1	col 2	col 3	col 4	col 5	col 6

Array A with active cell $(1, 1)$, which contains the entry $A_{1,1} = -5$.

Moving Heap Method

Algorithms for
Discrete
Random
Variables

Outline

Data
Structures and
Simple
Algorithms

Sums of
Independent
Random
Variables

Order
Statistics

Building the array of sums and probabilities

- Let the one-dimensional array s of length $n \cdot m$ hold the sums.
- Let the one-dimensional array called *Probs* of same length hold the corresponding probabilities for the sums.
- The first (smallest) sum to be placed in the first position of array s is $A_{1,1}$.
- After $A_{1,1}$ is removed, activate the cells with the next largest sums.

Moving Heap Method

Building the array of sums and probabilities

- Let the one-dimensional array s of length $n \cdot m$ hold the sums.
- Let the one-dimensional array called *Probs* of same length hold the corresponding probabilities for the sums.
- The first (smallest) sum to be placed in the first position of array s is $A_{1,1}$.
- After $A_{1,1}$ is removed, activate the cells with the next largest sums.

Moving Heap Method

Algorithms for
Discrete
Random
Variables

Outline

Data
Structures and
Simple
Algorithms

Sums of
Independent
Random
Variables

Order
Statistics

Building the array of sums and probabilities

- Let the one-dimensional array s of length $n \cdot m$ hold the sums.
- Let the one-dimensional array called *Probs* of same length hold the corresponding probabilities for the sums.
- The first (smallest) sum to be placed in the first position of array s is $A_{1,1}$.
- After $A_{1,1}$ is removed, activate the cells with the next largest sums.

Moving Heap Method

Building the array of sums and probabilities

- Let the one-dimensional array s of length $n \cdot m$ hold the sums.
- Let the one-dimensional array called *Probs* of same length hold the corresponding probabilities for the sums.
- The first (smallest) sum to be placed in the first position of array s is $A_{1,1}$.
- After $A_{1,1}$ is removed, activate the cells with the next largest sums.

Moving Heap Method

Algorithms for
Discrete
Random
Variables

Outline

Data
Structures and
Simple
Algorithms

Sums of
Independent
Random
Variables

Order
Statistics

- The next cells that may contain the next largest sums are $A_{1,2} = -3$ and $A_{2,1} = -2$.
- Since cells (1, 2) and (2, 1) are now “active,” the 1’s and 0’s outside the matrix A are reset to reflect this

row 5	1	1	0	0	0	
row 4 8						0
row 3 5						0
row 2 1	-2					1
row 1 -2	-5	-3				1
	-3	-1	2	6	8	
	col 1	col 2	col 3	col 4	col 5	col 6

Moving Heap Method

Algorithms for
Discrete
Random
Variables

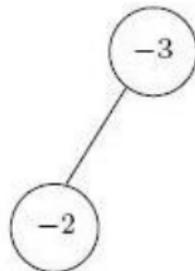
Outline

Data
Structures and
Simple
Algorithms

Sums of
Independent
Random
Variables

Order
Statistics

- The values $A_{1,2}$ and $A_{2,1}$ now form the minimum heap H , resulting in $H = \{A_{1,2}, A_{2,1}\} = \{-3, -2\}$.
- A minimum heap is a complete binary tree with the special ordering property that each parent node contains a value less than or equal to the values in its children's nodes.
- Because of this ordering property, the smallest value in a minimum heap will always be at the root.
- The next sum to be entered into the array of sums s is the root of the heap.



Moving Heap Method

Algorithms for
Discrete
Random
Variables

Outline

Data
Structures and
Simple
Algorithms

Sums of
Independent
Random
Variables

Order
Statistics

- Array A after $A_{1,2} = -3$ is processed and appended to s .
- Cell $(2, 1)$ is the only active cell. Candidates to become active are cells $(1, 3)$ and $(2, 2)$.

row 5	1	0	0	0	0	
row 4	8					0
row 3	5					0
row 2	1	-2				1
row 1	-2	-5	-3			0
	-3	-1	2	6	8	
	col 1	col 2	col 3	col 4	col 5	col 6

A diagram illustrating the Moving Heap Method. The grid shows values in cells. A thick vertical line is drawn between column 5 and column 6. A thick horizontal line is drawn between row 4 and row 5. The cell at row 2, column 1 (value -2) is highlighted in light pink. The cells at row 1, column 1 (value -5) and row 1, column 2 (value -3) are shaded with diagonal lines. An upward arrow points from the cell at row 1, column 2 to the cell at row 2, column 2. A rightward arrow points from the cell at row 1, column 2 to the cell at row 1, column 3.

Moving Heap Method

- Since row two contains the active cell (2, 1), and by design $A_{2,1} < A_{2,2}$, entry $A_{2,2}$ is not activated.
- Cell (1, 3) does become active, and its entry is $A_{1,3} = y_1 + x_3 = 0$.
- Since cells (1, 3) and (2, 1) are now “active,” the 1’s and 0’s outside the matrix A are reset to reflect this.

row 5		1	0	1	0	0	
row 4	8						0
row 3	5						0
row 2	1	-2					1
row 1	-2	-5	-3	0			1
		-3	-1	2	6	8	
		col 1	col 2	col 3	col 4	col 5	col 6

Moving Heap Method

Algorithms for
Discrete
Random
Variables

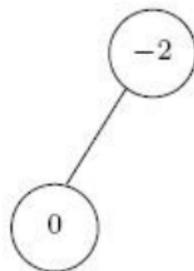
Outline

Data
Structures and
Simple
Algorithms

Sums of
Independent
Random
Variables

Order
Statistics

- The entry $A_{1,3}$ is inserted into the heap H , resulting in $H = \{A_{2,1}, A_{1,3}\} = \{-2, 0\}$.
- After the addition of $A_{1,3}$, the heap H is displayed below.
- The minimum element, $A_{2,1}$, is removed from the root of the heap and placed in the sum array s ; its corresponding probability is placed in the *Probs* array.



Moving Heap Method

Algorithms for
Discrete
Random
Variables

Outline

Data
Structures and
Simple
Algorithms

Sums of
Independent
Random
Variables

Order
Statistics

Continuing to move northeast through matrix A

row 5		0	0	1	0	0	
row 4	8						0
row 3	5						0
row 2	1	$\leftarrow 2$					0
row 1	-2	$\leftarrow 5$	$\leftarrow 3$	0			1
		-3	-1	2	6	8	
		col 1	col 2	col 3	col 4	col 5	col 6

A diagram illustrating the Moving Heap Method. The matrix A is shown with rows 1 to 5 and columns 1 to 6. The current position is at row 2, column 1, with a value of 1. The path of the moving heap is indicated by arrows: from row 2, column 1 to row 3, column 1 (up), and from row 2, column 1 to row 2, column 2 (right). The cells (row 2, column 1), (row 3, column 1), and (row 1, column 1) are shaded with diagonal lines. The cell (row 1, column 3) is shaded light pink. A thick black vertical line is drawn between column 5 and column 6.

Moving Heap Method

Algorithms for
Discrete
Random
Variables

Outline

Data
Structures and
Simple
Algorithms

Sums of
Independent
Random
Variables

Order
Statistics

row 5		1	1	1	0	0	
row 4	8						0
row 3	5	2					1
row 2	1	-2	0				1
row 1	-2	-5	-3	0			1
		-3	-1	2	6	8	
		col 1	col 2	col 3	col 4	col 5	col 6

Figure: Array A with active cells $(1, 3)$, $(2, 2)$, and $(3, 3)$.

Moving Heap Method

Algorithms for
Discrete
Random
Variables

Outline

Data
Structures and
Simple
Algorithms

Sums of
Independent
Random
Variables

Order
Statistics

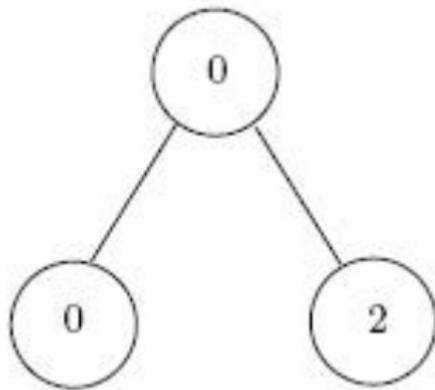


Figure: Heap H with entries $A_{1,3} = 0$, $A_{2,2} = 0$ and $A_{1,3} = 2$.

Moving Heap Method

Algorithms for
Discrete
Random
Variables

Outline

Data
Structures and
Simple
Algorithms

Sums of
Independent
Random
Variables

Order
Statistics

row 5		0	0	0	1	0	
row 4	8	5	7	10			0
row 3	5	2	4	7	11		1
row 2	1	-2	0	3	7	9	0
row 1	-2	-5	-3	0	4	6	0
		-3	-1	2	6	8	
		col 1	col 2	col 3	col 4	col 5	col 6

Figure: Array A with its seventeenth active cell (3, 4).

Moving Heap Method

- After twenty iterations of this process, s and $Probs$ arrays are

$$s = [-5, -3, -2, 0, 0, 2, 3, 4, 4, 5, 6, 7, 7, 7, 9, 10, 11, 13, 14, 16]$$

$$Probs = [0.03, 0.05, 0.015, 0.025, 0.02, 0.045, 0.01, 0.075, 0.06, 0.06, 0.04, 0.1, 0.03, 0.03, 0.02, 0.04, 0.09, 0.06, 0.12, 0.08]$$

- These are the same arrays constructed by using the moment generating function technique.
- The redundancies are removed from s and the appropriate probabilities are combined in $Probs$.

Sums of Independent Discrete Random Variables

Algorithms for
Discrete
Random
Variables

Outline

Data
Structures and
Simple
Algorithms

Sums of
Independent
Random
Variables

Order
Statistics

Summary

If supports of X and Y are:

- “small” (less than 100), use the **Brute Force Method**.
- finite, but “large” (greater than 100), use the **Moving Heap Sort Method**.
- infinite, use either the **Discrete Convolution Formula** or **Moment Generating Function Technique**.

Sums of Independent Discrete Random Variables

Summary

If supports of X and Y are:

- “small” (less than 100), use the **Brute Force Method**.
- finite, but “large” (greater than 100), use the **Moving Heap Sort Method**.
- infinite, use either the **Discrete Convolution Formula** or **Moment Generating Function Technique**.

Sums of Independent Discrete Random Variables

Algorithms for
Discrete
Random
Variables

Outline

Data
Structures and
Simple
Algorithms

Sums of
Independent
Random
Variables

Order
Statistics

Summary

If supports of X and Y are:

- “small” (less than 100), use the **Brute Force Method**.
- finite, but “large” (greater than 100), use the **Moving Heap Sort Method**.
- infinite, use either the **Discrete Convolution Formula** or **Moment Generating Function Technique**.

Sums of Independent Discrete Random Variables

Algorithms for
Discrete
Random
Variables

Outline

Data
Structures and
Simple
Algorithms

Sums of
Independent
Random
Variables

Order
Statistics

Example

Let the discrete random variable $S = X_1 + X_2 + \cdots + X_{150}$, where the X_i 's are independent, $\Pr(X_i = -1) = \Pr(X_i = 0) = \Pr(X_i = 1) = 1/3$, $i = 1, 2, \dots, 150$. Find a normal approximation to $\Pr(S = 5)$.

Instead of settling for approximations of the probabilities, APPL procedures, such as `Convolution`, can retrieve exact solutions.

Since the mean and variance of S are $\mu = 0$ and $\sigma^2 = 100$, using the normal PDF approximation we obtain

$$\frac{1}{20} \frac{\sqrt{2}e^{-1/8}}{\sqrt{\pi}} \cong 0.03521.$$

Sums of Independent Discrete Random Variables

Algorithms for
Discrete
Random
Variables

Outline

Data
Structures and
Simple
Algorithms

Sums of
Independent
Random
Variables

Order
Statistics

Example

Let the discrete random variable $S = X_1 + X_2 + \cdots + X_{150}$, where the X_i 's are independent, $\Pr(X_i = -1) = \Pr(X_i = 0) = \Pr(X_i = 1) = 1/3$, $i = 1, 2, \dots, 150$. Find a normal approximation to $\Pr(S = 5)$.

Instead of settling for approximations of the probabilities, APPL procedures, such as `Convolution`, can retrieve exact solutions.

Since the mean and variance of S are $\mu = 0$ and $\sigma^2 = 100$, using the normal PDF approximation we obtain

$$\frac{1}{20} \frac{\sqrt{2}e^{-1/8}}{\sqrt{\pi}} \cong 0.03521.$$

Sums of Independent Discrete Random Variables

Algorithms for
Discrete
Random
Variables

Outline

Data
Structures and
Simple
Algorithms

Sums of
Independent
Random
Variables

Order
Statistics

The APPL code:

```
X := [[1 / 3, 1 / 3, 1 / 3], [-1, 0, 1],  
      ["Discrete", "PDF"]];  
S := ConvolutionIID(X, 150)
```

yields the exact PDF for S . The statement $\text{PDF}(S, 5)$ returns

$$\Pr(S = 5) = \frac{160709987007649212790999852367465829596098558279031212787052332840770}{4567759074507740406477787437675267212178680251724974985372646979033929},$$

which is approximately 0.03518.

`ConvolutionIID` is a procedure that uses `Convolution` to sum 2 or more independent and identically distributed random variables.

Order Statistics for Discrete RVs: Taxonomy

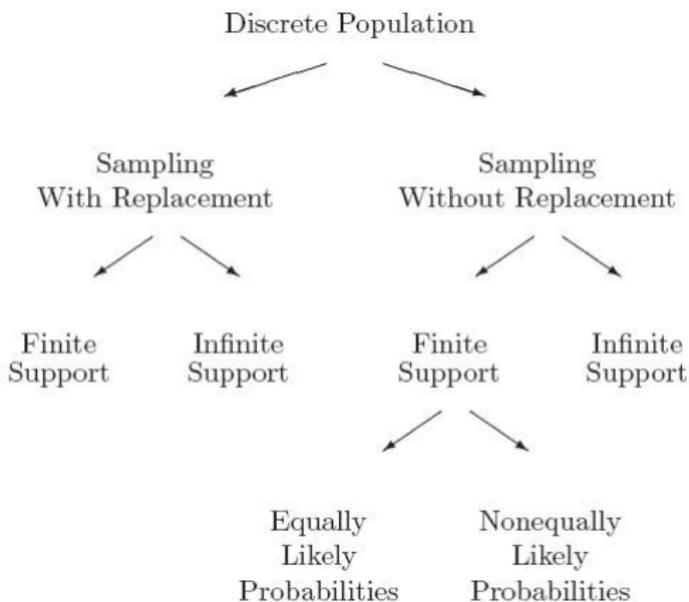


Figure: Categorization of discrete order statistics by sampling convention, support, and probability distribution.

APPL's Order Statistics Procedure

Algorithms for
Discrete
Random
Variables

Outline

Data
Structures and
Simple
Algorithms

Sums of
Independent
Random
Variables

Order
Statistics

The APPL procedure `OrderStat(X, n, r, ["wo"])` has three required arguments:

- a random variable X ,
- the number of items n randomly drawn from the population with PDF $f_X(x)$, and
- the index r of the desired order statistic.
- An optional fourth argument "wo" can be specified to indicate that the items are drawn from the population *without* replacement.

The output of the algorithm is $f_{X_{(r)}}(x)$, the PDF of the r th order statistic, where n items have been sampled either with or without replacement from a population with PDF $f_X(x)$.

Sampling Without Replacement, Finite Support

Algorithms for
Discrete
Random
Variables

Outline

Data
Structures and
Simple
Algorithms

Sums of
Independent
Random
Variables

Order
Statistics

If the population with PDF $f_X(x)$ has **equally likely probability values**, then the PDF of the r th order statistic when n items are sampled is

$$f_{X_{(r)}}(x) = \frac{\binom{x-1}{r-1} \binom{N-x}{n-r}}{\binom{N}{n}} \quad x = r, r+1, \dots, r+N-n.$$

Sampling Without Replacement, Finite Support

If $f_X(x)$ has **nonequally likely probability values**, then there are three cases to consider when computing the PDF of the r th order statistic:

- 1 One item is sampled: $n = 1$. The PDF of the r th order statistic is the same as the population PDF; i.e.,
$$f_{X_{(r)}}(x) = f_X(x) \text{ for } r = 1, 2, \dots, N.$$
- 2 The entire population is sampled: $n = N$. The PDF of the r th order statistic is $f_{X_{(r)}}(x) = 1$ for $x = r$.
- 3 More than one item, but not the entire population, is sampled: $n = 2, 3, \dots, N - 1$. This non-trivial case required us to write additional procedures `NextCombination` and `NextPermutation`.

Sampling Without Replacement, Finite Support

If $f_X(x)$ has **nonequally likely probability values**, then there are three cases to consider when computing the PDF of the r th order statistic:

- 1 One item is sampled: $n = 1$. The PDF of the r th order statistic is the same as the population PDF; i.e.,
$$f_{X_{(r)}}(x) = f_X(x) \text{ for } r = 1, 2, \dots, N.$$
- 2 The entire population is sampled: $n = N$. The PDF of the r th order statistic is $f_{X_{(r)}}(x) = 1$ for $x = r$.
- 3 More than one item, but not the entire population, is sampled: $n = 2, 3, \dots, N - 1$. This non-trivial case required us to write additional procedures `NextCombination` and `NextPermutation`.

Sampling Without Replacement, Finite Support

If $f_X(x)$ has **nonequally likely probability values**, then there are three cases to consider when computing the PDF of the r th order statistic:

- 1 One item is sampled: $n = 1$. The PDF of the r th order statistic is the same as the population PDF; i.e.,
$$f_{X_{(r)}}(x) = f_X(x) \text{ for } r = 1, 2, \dots, N.$$
- 2 The entire population is sampled: $n = N$. The PDF of the r th order statistic is $f_{X_{(r)}}(x) = 1$ for $x = r$.
- 3 More than one item, but not the entire population, is sampled: $n = 2, 3, \dots, N - 1$. This non-trivial case required us to write additional procedures `NextCombination` and `NextPermutation`.

Sampling Without Replacement, Finite Support

Further exploration of Case 3:

- (a) The first lexicographical combination of n items sampled from the sequence of integers 1 to N is formed; it is $\{1, 2, \dots, n\}$.
- (b) Given a combination consisting of n distinct integers, the algorithm generates all possible permutations of that combination and their corresponding probabilities.
- (c) After the probability of each permutation generated in step (b) is computed, each permutation is “rewritten” lexicographically to determine the corresponding order statistic probabilities.
- (d) After all $n!$ permutations of a given combination are exhausted, the procedure `NextCombination` determines the next lexicographical combination. Steps (b) and (c) are repeated.

Sampling Without Replacement, Finite Support

Further exploration of Case 3:

- (a) The first lexicographical combination of n items sampled from the sequence of integers 1 to N is formed; it is $\{1, 2, \dots, n\}$.
- (b) Given a combination consisting of n distinct integers, the algorithm generates all possible permutations of that combination and their corresponding probabilities.
- (c) After the probability of each permutation generated in step (b) is computed, each permutation is “rewritten” lexicographically to determine the corresponding order statistic probabilities.
- (d) After all $n!$ permutations of a given combination are exhausted, the procedure `NextCombination` determines the next lexicographical combination. Steps (b) and (c) are repeated.

Sampling Without Replacement, Finite Support

Further exploration of Case 3:

- (a) The first lexicographical combination of n items sampled from the sequence of integers 1 to N is formed; it is $\{1, 2, \dots, n\}$.
- (b) Given a combination consisting of n distinct integers, the algorithm generates all possible permutations of that combination and their corresponding probabilities.
- (c) After the probability of each permutation generated in step (b) is computed, each permutation is “rewritten” lexicographically to determine the corresponding order statistic probabilities.
- (d) After all $n!$ permutations of a given combination are exhausted, the procedure `NextCombination` determines the next lexicographical combination. Steps (b) and (c) are repeated.

Sampling Without Replacement, Finite Support

Further exploration of Case 3:

- (a) The first lexicographical combination of n items sampled from the sequence of integers 1 to N is formed; it is $\{1, 2, \dots, n\}$.
- (b) Given a combination consisting of n distinct integers, the algorithm generates all possible permutations of that combination and their corresponding probabilities.
- (c) After the probability of each permutation generated in step (b) is computed, each permutation is “rewritten” lexicographically to determine the corresponding order statistic probabilities.
- (d) After all $n!$ permutations of a given combination are exhausted, the procedure `NextCombination` determines the next lexicographical combination. Steps (b) and (c) are repeated.

Sampling Without Replacement, Infinite Support

- The OrderStat algorithm computes the PDF of the minimum order statistic when at most $n = 2$ items are sampled without replacement from a discrete population with infinite support; $n \geq 3$ is an open research question.

The PDF of $X_{(1)}$ when $n = 2$ items are sampled is

$$f_{X_{(1)}}(x) = f_X(x) \left[\frac{S_X(x+1)}{1 - f_X(x)} + \sum_{y=x+1}^{\infty} \frac{f_X(y)}{1 - f_X(y)} \right] \quad x = 1, 2, \dots,$$

where $S_X(x)$ is the survivor function defined by $S_X(x) = \Pr(X \geq x)$.

Sampling With Replacement, Finite Support

Algorithms for
Discrete
Random
Variables

Outline

Data
Structures and
Simple
Algorithms

Sums of
Independent
Random
Variables

Order
Statistics

If the random variable X has finite support Ω , then without loss of generality we can assume that $\Omega = \{1, 2, \dots, N\}$.

The PDF of $X_{(r)}$ when n items are sampled with replacement from this finite population is given by

$$f_{X_{(r)}}(x) = \begin{cases} \sum_{w=0}^{n-r} \binom{n}{w} [f_X(1)]^{n-w} [S_X(2)]^w & x = 1 \\ \sum_{u=0}^{r-1} \sum_{w=0}^{n-r} \binom{n}{u, n-u-w, w} [F_X(x-1)]^u [f_X(x)]^{n-u-w} [S_X(x+1)]^w & x = 2, 3, \dots, N-1 \\ \sum_{u=0}^{r-1} \binom{n}{u} [F_X(N-1)]^u [f_X(N)]^{n-u} & x = N. \end{cases}$$

Sampling With Replacement, Infinite Support

Algorithms for
Discrete
Random
Variables

Outline

Data
Structures and
Simple
Algorithms

Sums of
Independent
Random
Variables

Order
Statistics

If the support of X is countably infinite, then the calculation of the PDF of $X_{(r)}$ is similar to the finite-support case.

The algorithm only works for distributions with infinite right-hand tails.

$$f_{X_{(r)}}(x) = \begin{cases} \sum_{w=0}^{n-r} \binom{n}{w} [f_X(1)]^{n-w} [S_X(2)]^w & x = 1 \\ \sum_{u=0}^{r-1} \sum_{w=0}^{n-r} \binom{n}{u, n-u-w, w} [F_X(x-1)]^u [f_X(x)]^{n-u-w} [S_X(x+1)]^w & x = 2, 3, \dots \end{cases}$$