Some Useful Formulas

May 7, 2012

1 Management Science

- The Fundamental Principle of counting (page 36): If there are a ways of choosing one thing, b ways of choosing a second after the first is chosen, ..., and z ways of choosing a last item after the earlier choices, then the total number of choice patterns is a*b*c*...*z.

- n factorial, written $n!$ is the product $n * (n-1) * (n-2) * \ldots * 2 * 1$.

- There are $\frac{(n-1)!}{2}$ distinct Hamiltonian circuits in a complete graph on n verticies (K_n).

- Traveling Salesperson

  Nearest-Neighbor Algorithm: (page 39) Starting from the home city, first visit the nearest city, then visit the nearest city that has not already been visited. We return to the start city when no other choice is available.

  Sorted-Edges Algorithm: (page 40) Start by sorting or arranging the edges of the complete graph in order of increasing cost (or, equivalently, arranging the intercity distances in order of increasing distance). Then at each stage select an edge that has not been previously chosen of least cost that

  1. Never requires that three used edges meet at a vertex (because a Hamiltonian circuit uses up exactly two edges at each vertes) and that
  2. Never closes up a circular tour that doesn't include all the vertices.

- Minimal Spanning Tree

  Kruskal's Algorithm (page 43): Add edges in order of cheapest cost so that no circuits form and so that every vertex belongs to some edge added.

- List Processing Algorithm

  Part I and Ready Task (page 69) The algorithm we use to schedule tasks is the **list-processing algorithm.** In describing it, we will call a task **ready** at a particular time if all its predecessors as indicated in the order-requirement digraph have been completed at that time. The algorithm works as follows: At a given time, assign to the lowest-numbered free processor the first task on the priority list that is *ready* at that time and that hasn't already been assigned to a processor.

  Part II (page 70) As the priority list is scanned from left to right to right to assign a task to a processor at a particular time, we pass over tasks that are not ready to find ones that are ready. If no task can be assigned in this manner, we keep one or more processors idle until such time that, reading the priority list from the left, there is a ready task not already assigned. After a task is assigned to a processor, we resume scanning the priority list, starting over at the far left, for unassigned tasks.

2 Statistics, Counting, and Probability

- Average (Mean) of a set of numeric data

$$\bar{x} = \frac{x_1 + x_2 + \ldots + x_n}{n}$$

- Standard Deviation of a set of numeric data

$$s_x = \sqrt{\frac{(x_1 - \bar{x})^2 + (x_2 - \bar{x})^2 + \ldots + (x_n - \bar{x})^2)}{(n-1)}}$$

- Pearson's Remarkable Correlation Coefficient

$$r = \frac{1}{(n-1)}((\frac{x_1 - \bar{x})}{s_x})(\frac{y_1 - \bar{y})}{s_y})+(\frac{x_2 - \bar{x})}{s_x})(\frac{y_2 - \bar{y})}{s_y})+\ldots+(\frac{x_n - \bar{x})}{s_x})(\frac{y_n - \bar{y})}{s_y}))$$

- Linear Regression:

$$\hat{y} = b + mx,$$

  where

  -
$$m = r\frac{s_y}{s_x}$$

  - and
$$b = \bar{y} - m\bar{x}$$

- Sampling Distribution of a Sample Proportion (from page 228): Choose an SRS of size n from a large population that contains population proportion p of successes. Let $\hat{p}$ be the **sample proportion** of successes,

$$\hat{p} = \frac{\text{count of successes in the sample}}{n}$$

then

Shape: For large ($n \geq 30$) sample sizes, the sampling distribution of $\hat{p}$ is *approximately normal*

Center: The *mean* of the sampling distribution of $\hat{p}$ is p

Spread: The *standard deviation* of the sampling distribution of $\hat{p}$ is

$$\sqrt{\frac{p(1-p)}{n}}$$

Where, in practice, we use $\hat{p}$ as an estimate for p.

- Margin of error and confidence intervals for sample proportions: The *margin of error (me)* for a 95% confidence interval for a sample proportion is given by

$$me = 2\sqrt{\frac{p(1-p)}{n}}$$

We report this in one of two ways:

  - Using the margin of error:

$$\hat{p} \pm 2\sqrt{\frac{p(1-p)}{n}}$$

or as an interval:

$$(\hat{p} - 2\sqrt{\frac{p(1-p)}{n}}, \hat{p} + 2\sqrt{\frac{p(1-p)}{n}})$$

Again, in these calculations, we use $\hat{p}$ as an estimate of p.

- Probability Rules (from page 253)

Rule 1 The probability P(A) of any event A satisfies $0 \leq P(A) \leq 1$

Rule 2 If S is the sample space in a probability model, then P(S) = 1

Rule 3 The **complement rule**: $P(A^c) = 1 - P(A)$

Rule 4 The **multiplication rule** for *independent* events:
P(A **and** B) = P(A) * P(B)

Rule 5 The *general* **addition rule**:
P(A **or** B) = P(A) + P(B) - P(A **and** B)

Rule 6 The *addition rule* for *disjoint* events:
P(A **or** B) = P(A) + P(B)

- Counting Rules (from page 261)

  Chosing k items from n distinct items

Order	Repetition is allowed	Repetition is not allowed
Order does matter	Rule A: n^k	Rule B: (permutations) $\frac{n!}{(n-k)!}$
Order does not matter	Rule C: $\frac{(n+k-1)!}{k!(n-1)!}$	Rule D: (combinations) $\frac{n!}{k!(n-k)!}$

- Mean and standard deviation of a Discrete Probability Model (pages 264 and 266)

Mean: Suppose that the possible outcomes $x_1, x_2, \ldots, x_n$ in a sample space S are numbers and that p_j is the probability of outcome x_j. The **mean** μ **of a discrete probability model** (sometimes called the **mathematical expectation** is

$$\mu = x_1 p_1 + x_2 p_2 + \ldots + x_n p_n$$

Standard Deviation: with the same notation, the **standard deviation** σ **of a discrete probability model** with mean μ is

$$\sigma = \sqrt{(x_1 - \mu)^2 p_1 + (x_2 - \mu)^2 p_2 + \ldots + (x_n - \mu)^2 p_n}$$

- The Central Limit Theorem (page 268):

  Draw an SRS of size n from any large population with mean μ and finite standard deviation σ. Then

  - The mean of the sampling distribution of $\bar{x}$ is μ.
  - The standard deviation of the sampling distribution of $\bar{x}$ is $\frac{\sigma}{\sqrt{n}}$
  - The **Central Limit Theorem** says that the sampling distribution of $\bar{x}$ is approximately normal with the sample size n is large ($n > 30$).

3 Game Theory

3.1 A procedure for finding the value of a Two-Person Total-Conflict Game using mixed strategies

The following outlines the procedure given in section 15.2 for finding the value of a two-person total-conflict game with at least one player (Player I in this example) employing a mixed strategy:

We are given a **playoff matrix** with Player I's pure strategies (the following assums two of them) as rows, and Player II's pure strategies (again, we assume two of them) in columns.

Step 1: We assume that Player I adopts one pure strategy with probability p (we assume further that this is the pure strategy associated with the first row of the payoff matrix) and the second with probably $(1-p)$ (the pure strategy associated with the second row of the payoff matrix). Our purpose first is to find the probability p which best serves Player I.

Step 2: Calculate the **expected value** of Player I's mixed strategy against Player II's first pure strategy (in column 1). That is, we calculate

$$E_1 = p(\text{payoff value in first row, first column}) + (1-p)(\text{payoff value in second row, first column})$$

Step 3: We do the same for Player II's second pure strategy (column 2). Calculate the **expected value** of Player I's mixed strategy against Player II's second pure strategy. That is, we calculate

$$E_2 = p(\text{payoff value in first row, second column}) + (1-p)(\text{payoff value in second row, second column})$$

Step 4: We now calculate the optimum value of p for Player I by setting $E_1 = E_2$ and solving for p.

Step 5: Finally, we find the **value of the game** by finding the expected value E_1 for the value of p found in step 4. This should, of course, be the same value found by plugging in p into the equation for E_2.