The Minimum of n Independent Normal Distributions

Joshua E. Hill

e-mail: josh-math@untruth.org

Initial Release: April 12, 2010
Current Revision: August 20, 2011

Problem: We have a race of n runners. What is the probability that a particular runner will win?

Each runner's event time can be viewed as a random variable, which we'll assume is distributed normally with a runner-specific mean and standard deviation for their time. The j th runner's time is distributed as $X_j \sim \text{NormalDistribution}\left(\mu_j, \sigma_j^2\right)$ (that is, normally distributed with mean μ_j and standard deviation σ_j).

First, let's examine the $n = 2$ case, that is the chance in a two person race that the first runner wins. We want $\Pr(X_0 < X_1)$ where X_0 and X_1 are distributed via different normal distributions (assumed independent). This is equivalent to $\Pr((X_0 - X_1) < 0)$, so you're subtracting two normal distributions (or adding X_0 with $-X_1$, it doesn't matter) which results in another normal distribution. $(X_0 - X_1) \sim \text{NormalDistribution}(\mu_0 - \mu_1, \sigma_0^2 + \sigma_1^2)$, so you get a simple calculation (an integral conducted via numerical methods or a table lookup) to figure what $\Pr((X_0 - X_1) < 0)$ is.

For more than two runners, it becomes a bit harder.

The way one would actually, in practice, solve this problem is to program a simulation and run many thousands of rounds of simulation given your particular data.

For a proper mathematical solution, the probability for player 0 to win in a race with $n + 1$ runners total is $\Pr(X_0 < Y)$, where

$$Y = \min_{1 \le j \le n} X_j$$

That is, Y is the distribution of the smallest time from the remaining n runners.

To find the distribution of Y, we first look for its cumulative distribution function (CDF). In the standard setting, we would follow this up by taking the derivative of the CDF to get the PDF, but we don't proceed in this way.

We start with some results for a single variable; we'll write the normal complementary cumulative distribution function as F_c.

$$\begin{aligned}\Pr(X_j > a) &= F_c(a)\\ &= 1 - \Pr(X_j \leq a)\\ &= 1 - \frac{1}{2}\left(1 + \operatorname{erf}\left(\frac{x-\mu_j}{\sigma_j\sqrt{2}}\right)\right)\\ &= \frac{1}{2}\left(1 - \operatorname{erf}\left(\frac{x-\mu_j}{\sigma_j\sqrt{2}}\right)\right)\\ &= \frac{1}{2}\operatorname{erfc}\left(\frac{a-\mu_j}{\sigma_j\sqrt{2}}\right)\end{aligned}$$

Where erfc is the complementary error function, defined as

$$\operatorname{erfc}(x) = 1 - \operatorname{erf}(x) = 1 - \frac{2}{\sqrt{\pi}}\int_0^x e^{-t^2}\,dt = \frac{2}{\sqrt{\pi}}\int_x^\infty e^{-t^2}\,dt$$

The probability $Y \leq a$ can be more easily viewed as the probability of the complement; that is, what is the chance that every $X_j (j \geq 1)$, is strictly greater than a. This will be in terms of the complementary cumulative distribution function for the distribution X_j, which we'll call $F_{c,j}$. We can define the cumulative distribution function for Y:

$$\begin{aligned}\Pr(Y \leq a) &= 1 - \Pr(X_1 > a \text{ and } X_2 > a \text{ and } \ldots \text{ and } X_n > a)\\ &= 1 - \prod_{j=1}^{n}\Pr(X_j > a) \qquad \text{(the distributions are independent)}\\ &= 1 - \prod_{j=1}^{n} F_{c,j}(a)\\ &= 1 - \prod_{j=1}^{n}\left(\frac{1}{2}\operatorname{erfc}\left(\frac{a-\mu_j}{\sigma_j\sqrt{2}}\right)\right)\\ &= 1 - \prod_{j=1}^{n}\frac{1}{\sqrt{\pi}}\int_{\frac{a-\mu_j}{\sigma_j\sqrt{2}}}^{\infty} e^{-t^2}\,dt\\ &= 1 - \prod_{j=1}^{n}\frac{1}{\sigma_j\sqrt{2\pi}}\int_a^\infty e^{-\left(\frac{t-\mu_j}{\sigma_j\sqrt{2}}\right)^2}\,dt\\ &= 1 - \frac{1}{\left(\prod_{j=1}^{n}\sigma_j\right)\left(\sqrt{2\pi}\right)^n}\int_a^\infty\cdots\int_a^\infty e^{-\frac{1}{2}\sum_{j=1}^{n}\left(\frac{t_j-\mu_j}{\sigma_j}\right)^2}\,dt_1\ldots dt_n\end{aligned}$$

From this last form, we can phrase this in terms of a multivariate normal distribution, with a covariance matrix

$$S = \left(\sigma_i^2 \delta_{i,j}\right)_{i,j} = \begin{pmatrix} \sigma_1^2 & & & & 0 \\ & \sigma_2^2 & & & \\ & & \ddots & & \\ & & & \sigma_{n-1}^2 & \\ 0 & & & & \sigma_n^2 \end{pmatrix}$$

With this convention, taking

$$\mathbf{t} = \begin{pmatrix} t_1 \\ \vdots \\ t_n \end{pmatrix} \text{ and } \boldsymbol{\mu} = \begin{pmatrix} \mu_1 \\ \vdots \\ \mu_n \end{pmatrix}$$

$$\begin{aligned}
\Pr(Y \leq a) &= 1 - \frac{1}{\left(\prod_{j=1}^n \sigma_j\right)\left(\sqrt{2\pi}\right)^n} \int_a^\infty \cdots \int_a^\infty e^{-\frac{1}{2}\sum_{j=1}^n \left(\frac{t_j - \mu_j}{\sigma_j}\right)^2} \, dt_1 \ldots dt_n \\
&= 1 - \frac{1}{\sqrt{\det S}\left(\sqrt{2\pi}\right)^n} \int_{(a,\infty)^n} e^{-\frac{1}{2}(\mathbf{t}-\boldsymbol{\mu})S^{-1}(\mathbf{t}-\boldsymbol{\mu})^{\mathrm{T}}} \, d\mathbf{t} \\
&= 1 - \int_{(a,\infty)^n} f(\mathbf{t}) \, d\mathbf{t}
\end{aligned}$$

where $f(\mathbf{t})$ is the PDF for this multivariate normal distribution

$$f(\mathbf{t}) = \frac{1}{\sqrt{\det S}\left(\sqrt{2\pi}\right)^n} e^{-\frac{1}{2}(\mathbf{t}-\boldsymbol{\mu})S^{-1}(\mathbf{t}-\boldsymbol{\mu})^{\mathrm{T}}}$$

We actually need $\Pr(Y > a)$, which, by the above, is $\int_{(a,\infty)^n} f(\mathbf{t}) \, d\mathbf{t}$

We'll refer to the PDF of X_0 as $g(s) = \frac{1}{\sqrt{2\pi\sigma_0^2}} e^{-\frac{1}{2}\left(\frac{(s-\mu_0)^2}{\sigma_0^2}\right)}$. We want to calculate

$$\begin{aligned}
\Pr(X_0 < Y) &= \int_{-\infty}^{\infty} \Pr(X_0 = s) \Pr(Y > s) \, ds \\
&= \int_{-\infty}^{\infty} g(s) \int_{(s,\infty)^n} f(\mathbf{t}) \, d\mathbf{t} \, ds \\
&= \int_{-(\infty,\infty)} \int_{(s,\infty)^n} g(s) f(\mathbf{t}) \, d\mathbf{t} \, ds
\end{aligned}$$

As may be clear from the above, multiplying the PDFs from independent normal distributions gives a multivariate normal distribution, so the product of f and g yields another multivariate distribution in one additional variable; the covariance matrix would be

$$W = \left(\sigma_i^2 \delta_{i,j}\right)_{i,j} = \begin{pmatrix} \sigma_0^2 & & & & 0 \\ & \sigma_1^2 & & & \\ & & \ddots & & \\ & & & \sigma_{n-1}^2 & \\ 0 & & & & \sigma_n^2 \end{pmatrix}$$

With this convention, taking

$$\mathbf{v} = \begin{pmatrix} v_0 \\ v_1 \\ \vdots \\ v_n \end{pmatrix} \text{ and } \boldsymbol{\mu}_f = \begin{pmatrix} \mu_0 \\ \mu_1 \\ \vdots \\ \mu_n \end{pmatrix}$$

and we get the PDF for this new distribution:

$$h(\mathbf{v}) = \frac{1}{\sqrt{\det W}\left(\sqrt{2\pi}\right)^{n+1}} e^{-\frac{1}{2}\left(\mathbf{v}-\boldsymbol{\mu}_f\right) W^{-1} \left(\mathbf{v}-\boldsymbol{\mu}_f\right)^{\mathrm{T}}}$$

so our final probability is:

$$Pr\,(X_0 < Y) = \int_{(-\infty,\infty)} \int_{(s,\infty)^n} h(s, t_1, \ldots, t_n)\, d\mathbf{t} ds$$

That's a wonderful clear statement conceptually, but doing calculations using this multivariate case is impractical. To do calculations, it is much preferable to stop much much sooner in the calculation:

$$\begin{aligned} \Pr(X_0 < Y) &= \int_{-\infty}^{\infty} \Pr(X_0 = s) \Pr(Y > s)\; ds \\ &= \int_{-\infty}^{\infty} g\,(s) \prod_{j=1}^{n} F_{c,j}(s)\, ds \\ &= \int_{-\infty}^{\infty} \frac{1}{\sqrt{2\pi\sigma_0^2}} e^{-\frac{1}{2}\left(\frac{(s-\mu_0)^2}{\sigma_0^2}\right)} \prod_{j=1}^{n} \left(\frac{1}{2}\operatorname{erfc}\left(\frac{s-\mu_j}{\sigma_j\sqrt{2}}\right)\right) ds \end{aligned}$$

The Mathematica code that does this calculation is provided in Listing 1.

Listing 1: Direct Calculation Approach

```
WinProb[playerList_] := Module[{u, v},
  u = PDF[playerList[[1]], x];
  v = Product[Integrate[PDF[playerList[[j]], t],
          {t, x, Infinity}],
      {j, 2, Length[playerList]}];
  NIntegrate[u*v, {x, -Infinity, Infinity}]
  ]
```

Here, we simply pass in a list containing the player time distribution (the analysis above only supports passing in normal distributions!).

To confirm that this works correctly, we can also create a routine to do this in simulation, provided in Listing 2.

Listing 2: Simulation Approach

```
WinProbSim[playerList_, rounds_] :=
 Module[{wins = 0, cntr, curRound},
  For[cntr = 0, cntr < rounds, cntr++,
   curRound = Map[RandomReal, playerList];
   If[curRound[[1]] == Min[curRound], wins++]
   ];
  wins/rounds
  ]
```

Running this proceeds in the same way, other than the fact that we need to tell the routine how many rounds to use in the simulation.

The intuitive case (where all runners have exactly the same characteristics) work out as expected (in the n player case, player 0 has a probability of $\frac{1}{n}$ of winning.

If we instead examine a race between 20 runners (player 0 to 19) in the 100 meter dash, where all players have an even time standard deviation of 1 second (wildly high, of course). Runners 1 through 19 have an average event time of 9.58 seconds (the world record for this event). Runner 0's mean time is shown as an advantage from the central time of 9.58 seconds, so his mean race time varies from 9.58 (advantage is 0) to 4.58 (advantage is 4).

The code that implements this race is presented in Listing 3.

Listing 3: Race 1 (Calculated)

```
FirstRace = Table[NormalDistribution[9.58, 1],
                  {i, 1, 19}];
AdvData =
  Table[{adv, WinProb[Prepend[FirstRace,
     NormalDistribution[9.58 - adv, 1]]]},
       {adv, 0, 5, .05}];
```

To run the same race in simulation, we instead use the code in Listing 4.

Listing 4: Race 1 (Simulation)

```
AdvSimData =
  Table[{adv, WinProbSim[Prepend[FirstRace,
     NormalDistribution[9.58 - adv, 1]],
        10000]}, adv, 0, 5, .05}];
```

When the simulation is run with 10,000 simulation rounds, the symbolic calculation is very close to the simulated result. Figure 1 depicts these results, with runner 0's advantage shown on the x-axis and the probability that runner 0 wins shown on the y-axis.

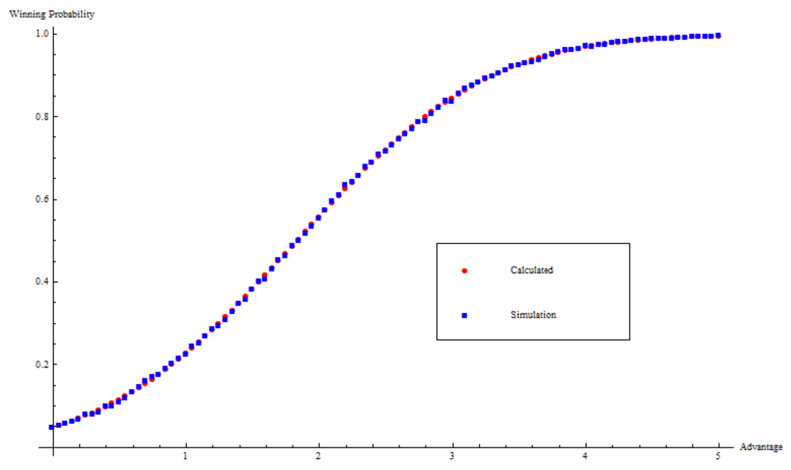

Figure 1: Race 1 Results

Let's now examine the situation where there is a wider array of runners. In this race, we'll include 10 runners, with all runners having an event time standard deviation of 0.5 seconds (still ridiculously high). Runners 1-9 have a mean time of 7.58 up to 11.58 seconds (they are each separated by 4/9 of a second). Runner 0's mean time is again shown as an advantage from the central time of 9.58 seconds, so it varies from 9.58 seconds (advantage is 0) to 4.58 seconds (advantage is 4). This advantage is shown on the x-axis. The probability that runner 0 wins is shown on the y-axis.

Direct computation (via numerical approximation) is accomplished using the code in Listing 5.

Listing 5: Race 2 (Calculated)

```
SecondRace = Table[NormalDistribution[9.58 + offset,
    .5], {offset, -2, 2, 4/9}];
SecondRaceAdvData =
  Table[{adv, WinProb[Prepend[SecondRace,
      NormalDistribution[9.58 - adv, 0.5]]]},
        {adv, 0, 5, .05}];
```

Simulation is accomplished using the code in Listing 6.

Listing 6: Race 2 (Simulation)

```
SecondRaceAdvSymData =
  Table[{adv, WinProbSim[Prepend[SecondRace,
      NormalDistribution[9.58 - adv, 0.5]],
        10000]}, {adv, 0, 5, .05}];
```

Again, these produce very similar results, as seen in Figure 2.

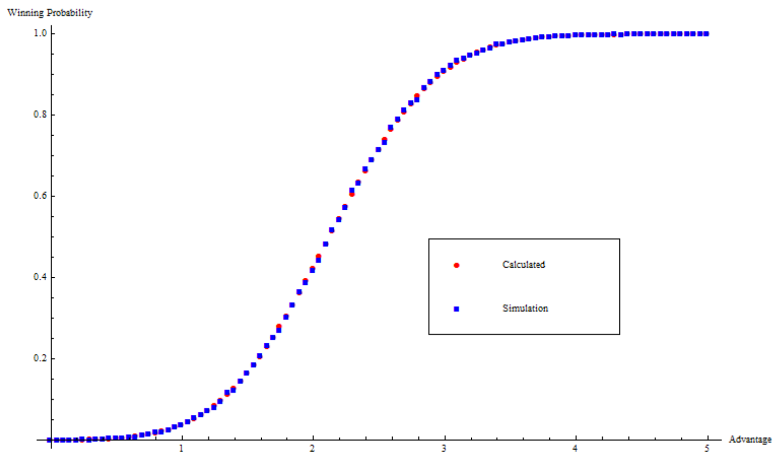

Figure 2: Race 2 Results

Colophon

The text of this document is typeset in Jean-François Porchez's wonderful Sabon Next typeface. Sabon Next is a modern (2002) revival of Jan Tschichold's 1967 Sabon typeface, which is in turn a adaptation of the classical (in all meanings) Garamond typeface, which dates from the early 16th century.

Equations are typeset using the MathTime Professional II (MTPro2) fonts, a font package released in 2006 by the great mathematical expositor Michael Spivak. These fonts are designed to work with the Times typeface, but they blend well with most classical fonts.

Source listings are typeset in Microsoft's Consolas, a monospace font with excellent readability.

XƎTEX was used to typeset the document, which is in turn offspring of Donald Knuth's profoundly important TEX. XƎTEX was selected in order to gain access to modern fonts without the trauma involved in converting them to a representation that pdfTeX could deal with. This approach makes most (though sadly, not all) OpenType features available, and sidesteps the traditional limit of 256 glyphs per font.

WinEdt 6 was used as an editor. Diagrams were produced in Mathematica.