Hyperparameter example: Micro-fractures on turbine blades

Your company manufactures gas turbines for power stations. One of the key performance characteristics is that the turbine blades do not, with high probability, develop micro-fractures beyond size X within an operation period T. There are 30 blades in a turbine. You test 18 turbines for the required period T, and perform an inspection of the blades using a sonic scanner. The inspection method is not foolproof: it has a 20% chance of failing to detect a fracture that is there. Your study identified 1 fracture. What is the probability that a turbine satisfies the performance requirement?

Contacting the manufacturer of the sonic scanner, you find that actually they are not so sure about this 20% failure rate: it is based on a study where 4 of 5 fractures were detected. How does this affect your estimate of the probability that a turbine satisfies the performance requirement?

If p is the probability that a turbine blade fails within the period T, then the probability that a turbine fails to meet the required performance specification P (TurbineFail) is given by:

P(TurbineFail) = 1-(1-P(BladeFail))^{30}

assuming that each blade are damaged independently of the others in a turbine. How can we estimate P(BladeFail)?

Two methods are presented below that yield the same posterior distribution.

Method 1: Turbine Blade Simulation

We tested 18 * 30 = 540 blades, of which 1 failed. That leads to an estimate of:

P(BladeFail) = Beta(1+1,540-1+1,1)

from estimation of the probability of success of a binomial trial. However, we are unsure of the number of blades *F* that actually are damaged.

F = NegBinomial(0.8,1+1) - 1

The total number of damaged blades we did not observe is calculated as the number of binomial trials to observe a certain number of successes. This formulae is a good approximation when F is much smaller than the population of 540 blades (as it is here) but wouldn't work well otherwise because the NegBinomial distribution has an infinite tail, and could produce more failures than we have blades.

We are also uncertain about the 20% failure rate (80% success rate) of detection P(detect):

P(detect) = Beta(4+1,5-4+1,1) = Beta(5,2,1)

Putting these altogether we get:

F = NegBinomial(Beta(5,2,1),1+1) - 1

P(BladeFail) = Beta(F+1,540-F+1,1)

$P(TurbineFail) = 1-(1- P(BladeFail))^{30}$

The object of this analysis was to determine P(BladeFail) and from that determine P(TurbineFail). A Bayesian estimate of P(BladeFail) required the parameter F, which was uncertain. The distribution for F is a hyperparameter. By using simulation to arrive at P(BladeFail) we have a natural way of integrating the extra uncertainty that F introduces into the calculation.

The links to the Turbine Blade Simulation software specific models are provided here:

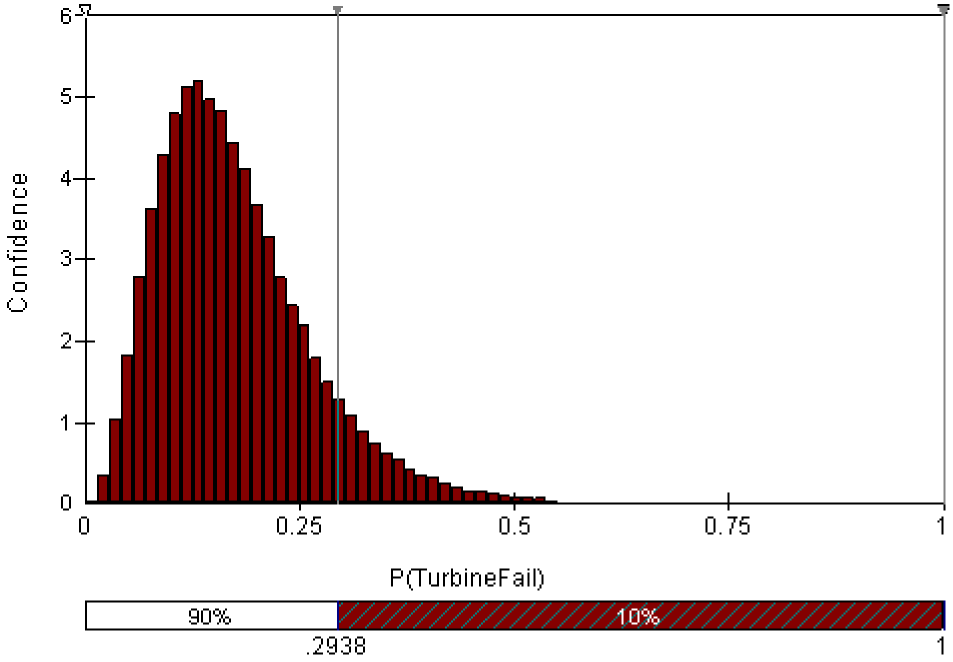
Turbine_blade_simulation

Here is teh result of the above model:

The graph shows that we are 90% confident that the probability a turbine fails within period T is less than 28%.

Turbine_blade_simulation

Here is the result of the above model:



The graph shows that we are 90% confident that the probability a turbine fails within period T is less than 29.4%.

Method 2: Turbine Blade Construction

We can construct a confidence distribution for P(BladeFail) = q as follows:

Uniform(0,1) prior:

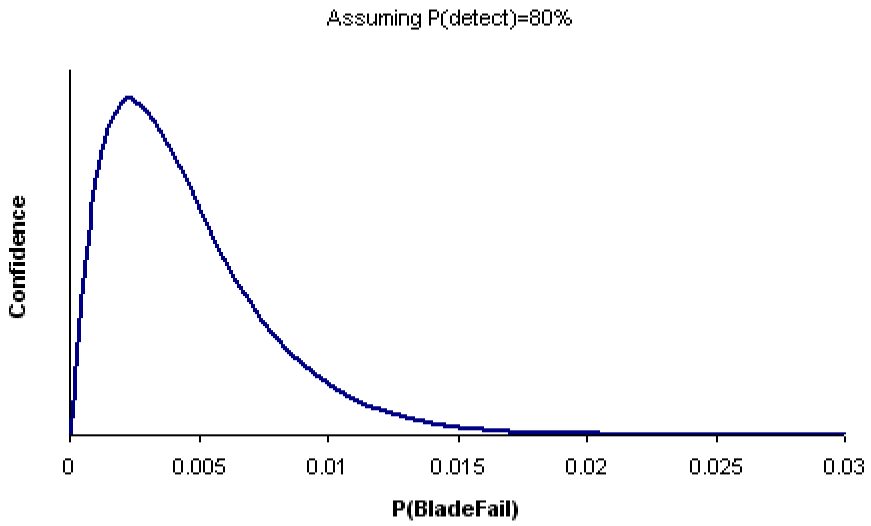
p(q) = 1

Binomial likelihood function:

=BINOMDIST(1,540,q*0.8,0)

which calculates the probability we would observe exactly (last parameter =0) 1 failed blades out of 540 when the probability of a blade failing (q) and then being observed (0.8) is q*0.8. Note that imbedding the observation probability 0.8 factor will work for all values of the problem, and is therefore superior to the NegBinomial method used in the simulation approach.

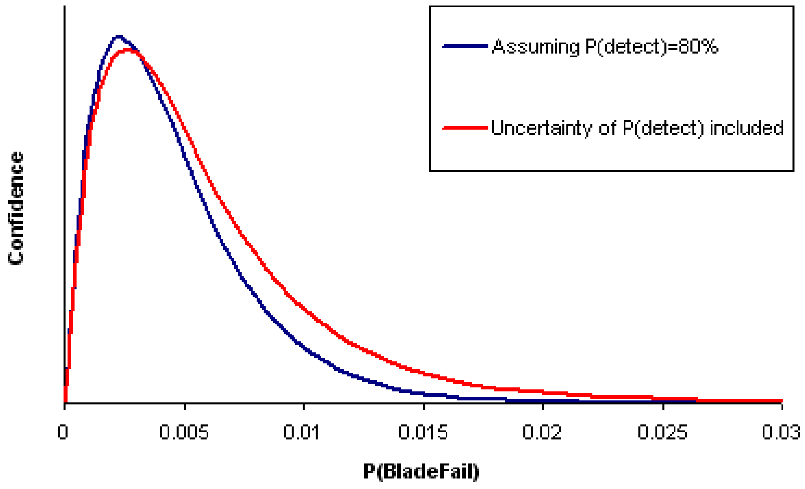
This leads to the following posterior distribution:



The analysis has not yet taken into account the uncertainty around P(detect), but we can do this with numerical integration. We can create a simulation for P(detect), as before:

P(detect) = Beta(4+1,5-4+1,1) = Beta(5,2,1)

and then link it to the BINOMDIST likelihood calculation. Then we calculate the posterior after this hyperparameter has been taken into account, by simply taking the mean of the simulated posterior density:



If you did not need a posterior distribution for P(BladeFail), the extra step of using the CB.GetForeStatFN(..,2) function is unnecessary: it is sufficient to simply run a simulation that samples from the Beta distribution for P(detect), constructs the General distribution for P(BladeFail), samples from that General distribution and calculates a value for P(TurbineFail), your output. All options are demonstrated in the accompanying model.

The links to the Turbine Blade Construction software specific models are provided here:

Turbine_blade_construction

In the model, Crystal Ball's Custom distribution was used to sample values from the uncertainty distribution.

Turbine_blade_construction

In the model, @Risk's General distribution was used to sample values from the uncertainty distribution.