Distribution Library

Provides inverse CDFs for the uncertainty calculations. Uses formulas scipy for some function calls.

# Libraries/Classes calling on Distribution Library

Uncertainty Summary

# Internal Library/Class dependencies

None

# External Library Dependencies

scipy

# Functions

invCDF() – Returns the value corresponding to the given probability and the input distribution and parameters

uniformInvCDF() – Returns the value corresponding to the given probability and the min and max values for the uniform distribution

triInvCDF() – Returns the value corresponding to the given probability and the min, mode, and max values for the triangular distribution

normInvCDF() – Returns the value corresponding to the given probability and the mean and standard deviation for the normal distribution

discInvCDF() – Returns the value corresponding to the given probability and a set of discrete values with associated probabilities

binomInvCDF() – Returns the value (number of successes) corresponding to the given probability and the number of trials for a binomial distribution

poisInvCDF() – Returns the value (number of occurrences) corresponding to the given probability and the return rate and number of intervals for a Poisson distribution

# Pseudo Code

Optional inputs are in italics

Begin Pseudocode

From scipy.stats import norm, binom, poisson

Define inverseCDF(distType, prob, params)

If distType is “uniform”

RV = uniformInvCDF(prob,params)

else if distType is “triangular”

RV = triInvCDF(prob,min,mode,max)

else if distType is “normal”

RV = normInvCDF(prob,mean,stdDev)

else if distType is “discrete”

RV = discInvCDF(prob,[[discVal0,discProb0],[discVal1,discProb1],…,[discValn,discProbn]])

else if distType is “binomial”

RV = binomInvCDF(prob,n,p,*loc*)

else if distType is “poisson”

RV = Poisson.ppf(prob,mu,loc=k)

Define uniformInvCDF(prob,min,max)

Formula

Return

End uniformInvCDF

Define triInvCDF(prob,min,mode,max)

If

Else

Return

End triInvCDF

Define normInvCDF(prob,mean,stdDev)

value = norm.ppf(prob,loc=mean,scale=stdDev

return value

end normInvCDF

Define discInvCDF(prob,[[discVal0,discProb0],[discVal1,discProb1],…,[discValn,discProbn]])

Check that the list of discVals and discProbs is sorted according to discVals, if not:

Sort according to discVals

Check that the sum of discProbs is one (within tolerance to account for precision error in decimal addition, if not:

Return error: Sum of probabilities is not equal to one

Loop through list of discVals and discProbs

If prob between 0 and discProb0 (inclusive of discProb0)

Return discProb0

Else if

Return discProbm

End discInvCDF

Define binomInvCDF(prob,n,p,*loc*)

Check if p between 0 and 1, if not:

Return error: p parameter must be between 0 and 1

Check if n is positive integer, if not:

Return error: n must be a positive integer

value = binom.ppf(prob,n,p,*x=loc*)

Note in almost all cases *loc* should be zero, if no *loc* variable is provided the x variable may be omitted

Return value

End binomInvCDF

Define poisInvCDF(prob,mu,k)

Poisson.ppf(prob,mu,loc=k)

End poisInvCDF