ES7023 - Assignment 5: Sensitivity analysis

An important part of being a good modeler is knowing how to perform sensitivity analysis (SA). Often the results of SA will simply confirm your intuition, but surprisingly often it will give you new insight into your model, even for fairly simple models. SA essentially comes down to understanding how model outputs respond to changes in model inputs. In lecture, we discussed some of the common goals (or 'settings') of sensitivity analysis (see Saltelli for more discussion):

1) Factor Prioritization (FP) - Identify which factors would provide the greatest reduction in output uncertainty if fixed. For example, will your modeling goals be better served by spending time and money to reduce uncertainty in factor A or B?
2) Factor Fixing (FF) - Identify which factors are non-influential, and therefore can be fixed without much change in the model. This is often the goal of 'screening' exercises, i.e. to screen out those factors that are not important
3) Factor Mapping (FM) - Identify which factors are mostly responsible for producing realizations of the output, Y, within different output classes (e.g., acceptable vs. non-acceptable levels of a contaminant).
4) Variance Cutting (VC) - Identify a minimum subset of factors that one should fix to achieve a prescribed reduction in output uncertainty.

There are several SA techniques, each with their own strengths and limitations. In general, simple techniques are suitable for most models, but more sophisticated techniques will be needed for models that are nonlinear, have several important factors, and/or have interactions between factors. As you know, this is often the case in environmental models. The suitability of a technique will also depend on the computational requirements of the models. For example, climate models take too long to run to make the more sophisticated SA techniques viable.

Here we will work with a simple example used in Saltelli et al. (2004). The model computes the minimum height above ground that a bungee jumper will reach, based on their starting height from the ground (H), mass (M), and the number of strands in the bungee cord (s). The idea is that we want an exciting jump (low minimum height) but without hitting the ground (minimum height > 0)! We don't know any of the input values perfectly. The model can be determined based on the solution to a linear oscillator equation as:

$$h_{min} = H - \frac{2 * 9.8 * M}{1.5 * s} \tag{1}$$

As you climb atop the platform, you make an educated guess about the input parameter values:

- The platform is roughly 50m from the ground, $\pm$ 10m.
- Last time you weighed yourself at 70.5kg, but that was three month ago, so let's say between 70.5kg $\pm$ 3.5kg.
- There are several choices of bungee coords to choose from, all having between 20-40 strands (or 30 $\pm$ 10 strands).

One-at-a-time (OAT) local SA:

A simple type of sensitivity analysis is to investigate each factor individually. One way to do this is to define a base case for all parameters, and then go one-at-a-time through each parameter and perturb it from its base value. This is called a local sensitivity analysis, since we are really only computing a partial derivative at a specific point. If the effect of one factor depends on the value of other factors, then a local SA can be misleading. But often it will provide at least a qualitative view of which factors are important and which are not (i.e. screening). Let's try that for our model. We will first define a range of uncertainty for each

parameter, and then test the effect of changing each parameter from baseline to its minimum and maximum value.

Problem 1:

Here you will conduct OAT local SA on the bungee jump model.

- Create the function to compute minimum height h_{min} from a jump. The function should look like `hmin = function (x) {...}` where x is a vector of three variables corresponding to H, M and s.
- The method is called "local SA" because it tests sensitivity around a specific design point. In our case this design point corresponds to the midpoint of our estimate of the input parameters: $H = 50$, $M = 70.5$ and $s = 30$. It is also a "One-at-A-Time" method because we test the sensitivity of each variable while keeping the others constant. Conduct OAT local SA of the bungee jump model by testing the effect of high and low values of each input parameters (based on the $\pm$ ranges described previously), while keeping the other parameters at their midpoint values.
- Make a barplot of the OAT local SA. Comment on the results.
- Repeat the same analysis using a different design point, but the same parameter ranges (i.e change the midpoint but keep range the same). Now try with the same design point ($H = 50$, $M = 70.5$ and $s = 30$) but different parameter ranges. Show and comment on the results.

One-at-a-time (OAT) global SA:

For many models, we may expect that the effect of uncertainty in a factor will depend on the values of (i.e. interact with) other factors, and so we will want to find a better measure of sensitivity than the local measure. A useful technique for this is the method of Morris. The Morris method is still an OAT approach, which means it takes only a few computations. It is described in detail in Saltelli et al. (2004). Here I have written a function in R to perform the Morris Method, following pp100-101 in the book. Note I assume that the distribution of each parameter is uniform, but this could be changed.

```
k = npars   #NUMBER OF PARAMETERS
r = 20          #NUMBER OF TIMES TO COMPUTE EFFECT FOR EACH PARAMETER
p = 4           #NUMBER OF POSSIBLE LEVELS FOR EACH PARAMETER (SHOULD BE EVEN)
delta = p/(2*(p-1)) #INCREMENT TO ADJUST PARAMETER VALUE BY

base.samp = function(p,n) {     #DEFINE FUNCTION TO SAMPLE FROM SET {0,1/(p-1),...1}
    x = (p-1-p/2) * runif(n)        #NEED TO LIMIT TO RANGE OF (0, 1-delta)
    round(x) / (p-1)
}

my.morris = function(k,r,p,delta,par.mins,par.maxs,fun) {
  #NOTE THE LAST INPUT IS THE MODEL NAME FOR SENSITIVITY ANALYSIS ANALYSIS

  par.range = par.maxs - par.mins
  effects = array(dim = c(k,r))

  for (r.ind in 1:r) {

    B = lower.tri(matrix(nrow = k+1,ncol=k)) * 1        #A LOWER TRIANGULAR MATRIX OF ONES
    xstar = base.samp(p,k)      #BASE VECTOR
    D=P=matrix(rep(0,k^2),nrow = k,ncol=k)
    diag(D)=sample(x=c(1,-1),k,replace=T) #DIAGONAL MATRIX WITH RANDOM 1 & -1 VALUES
    diag(P) = 1
    P = P[,sample(k)]
    J = matrix(1,nrow=k+1,ncol=k)   #A MATRIX OF ONES
    #SAMPLING MATRIX (EQ 4.7 SALTELLI 2004)
    Bstar =  (t(xstar*t(J)) + (delta/2) * ((2*B - J)%*%D + J)) %*% P
```

```
    y = numeric(length=(k+1))   #VECTOR TO HOLD MODEL OUTPUT
    for (i in 1:(k+1)) y[i] = fun(par.mins + par.range * Bstar[i,] )
    for (i in 1:k) {
      #FIND WHICH PARAMETER CHANGED AND WHETHER UP OR DOWN
        par.change = Bstar[i+1,] - Bstar[i,]
        i2 = which(par.change != 0)
        effects[i2,r.ind] = (y[i+1] - y[i]) * (1 - 2 * (par.change[i2] < 0))
    }
  }
  t(effects)        #RETURN THE r x k ARRAY OF COMPUTED EFFECTS
}
```

Problem 2

1. Now run the Morris method for the bungee model you developed, and plot the mean absolute effect and standard deviation of the effect, as shown in lecture slides. If you have coded your `hmin` function as `hmin = function (x) {...}` where x is a vector of inputs, then you should be able to run the code shown below. You can then compute the average absolute effect (use `abs()` function for absolute value) and standard deviation of effect for each input parameter. Provide interpretation of the results.

```
set.seed(42)     #SET SEED FOR PSEUDO-RANDOM NUMBER GENERATOR

k = npars   #NUMBER OF PARAMETERS
r = 20           #NUMBER OF TIMES TO COMPUTE EFFECT FOR EACH PARAMETER
p = 4            #NUMBER OF POSSIBLE LEVELS FOR EACH PARAMETER (SHOULD BE EVEN)
delta = p/(2*(p-1)) #INCREMENT TO ADJUST PARAMETER VALUE BY

par.mins = c(40, 67, 20)            #DEFINE RANGE FOR PARAMETERS
par.maxs = c(60, 74, 40)

effects = my.morris (k,r,p,delta,par.mins,par.maxs,hmin)
```

2. Now rerun the Morris method for the bungee model but with different ranges of input parameters. Plot the results for various ranges on a single plot. How are the values of average absolute effect and standard deviation of effect changed if the range and / or midpoint of the input parameter distributions changed?

Variance-based global sensitivity analysis

Variance-based SA techniques offer the ability to provide a well-defined measure of sensitivity: the amount (i.e. fraction) of variance in output that is reduced if we were to know the factor perfectly. Again, the trick is to get a global measure by sampling for different values of the parameter space. But in this case we will not vary each one individually, but vary all of them together.

The implementation of this variance-based SA method is that described in Saltelli 2004 P. 125-127.

```
set.seed(42)

my.vsa=function(fun,par.mins,par.maxs,nrun){
  par.range = par.maxs - par.mins
  k=length(par.range)
  M = matrix(runif(nrun*k),nrow=nrun,ncol=k)      #CREATE U(0,1) n x k MATRIX
  #TRANSFORM TO FIT THE RANGE OF EACH PARARAMETER
  for (i in 1:k) M[,i] = M[,i] * par.range[i] + par.mins[i]
```

```
  #Now we are going to make a different input matrix, M2, the same way
  M2 = array(runif(nrun*k),dim=c(nrun,k))      #CREATE U(0,1) n x k MATRIX
  for (i in 1:k) M2[,i] = M2[,i] * par.range[i] + par.mins[i]

  #Now we are going to make K different matrices, where in the jth matrix,
  #all parameters are taken from M2 except parameter j is taken from M
  NJ.list = list()
  for (j in 1:k) {
    temp = M2
    temp[,j] = M[,j]
    NJ.list[[j]] = temp
  }

  #Now, to compute the first-order effects (S) of factor J
  #we need to compute the values UJ, as discussed in class
  S = numeric(length=k)
  ST = numeric(length=k)
  y1 = y2 = y3 = numeric(length = nrun)
  for (i in 1:nrun) {
      y1[i] = fun(M[i,])
      y2[i] = fun(M2[i,])
  }
 #Now to compute the expected value (EY) and total variance (VY) of Y,
 #for which we will use M2
  EY=mean(y1)
  VY=sum(y1*y1)/(nrun-1)-EY^2

  for (j in 1:k) {
    NJ = NJ.list[[j]]
    for (i in 1:nrun) y3[i] = fun(NJ[i,])
    UJ = sum(y1*y3) / (nrun-1)   #here everything but factor j is resampled
    UJ2 = sum(y2*y3) / (nrun-1)  #here only factor j is resampled
    S[j] = (UJ - EY^2) / VY
    ST[j] = 1.0 - (UJ2 - EY^2) / VY
  }

  data.frame(Main=S,Total=ST)
}
```

Problem 3

1. Run the variance-based sensitivity analysis method for the bungee model using `nrun = 10000`. Make a figure showing both the main and total effects and discuss whether results are consistent with the results from the Morris Method for the same parameter ranges. Note: The main and total effects should generally both be between 0 and 1, and the main effect should be below the total effect. However, for computational reasons, sometimes factors that should have an effect close to zero will have slightly negative values for their main and/or total effects, and sometimes the main effect will appear to be slightly higher than the total effect. Don't worry if you see this in your results.
2. Now randomly sample 10,000 values of your three input parameters, and use your bungee model to calculate the min height associated with each sample. Plot the distribution of min heights reached. Flip the axis so the height is on the y-axis, and add a horizontal line at zero. Based on the uncertainty in input parameters, what portion of jumps lead to catastrophe?
3. Based on the results of your sensitivity analysis, how would you suggest reducing the probability of

catastrophic jumps? Which factor would you prioritize for measurement or reducing the range of uncertainty?

Optional: The value 1.5 in the bungee model is the elastic constant of one bungee strand. Having done some material testing, you discover that it too is quite variable. In fact it follows a uniform distribution ranging from 1.25 to 1.75. Write an `hmin_2` function that also includes the elastic constant as a random variable. Rerun 10000 sample jumps also including uncertainty in the elastic constant. What portion of the jumps lead to catastrophe and how does this compare to your results from problem 3 part 2 above? Run variance based sensitivity analysis on this new model and provide a brief commentary.