Sensitivity analysis of deterministic models through Latin hypercube sampling: Solutions

John M. Drake & Pejman Rohani

Exercise 1. In R, the function `cor` may be used to calculate the correlation between two sets of numbers. Write a script to determine which variable is most highly correlated with infections prevented.

To solve this exercise we first need to perform the analysis in the handout.

Here are the functions that we need.

```
> lambdaS <- function(betaSU,YSU,betaST,YST,pSU,YRU,pST,YRT,N){
+   (betaSU*YSU+betaST*YST+pSU*betaSU*YRU+pST*betaST*YRT)/N
+ }
> lambdaR <- function(betaRU,YRU,betaRT,YRT,N) (betaRU*YRU+betaRT*YRT)/N
```

Here is the function to evaluate the model.

```
> hiv.model.sf <- function(t, x, params){
+   X <- x[1]
+   Y <- x[2]
+   YSU <- x[3]
+   YST <- x[4]
+   YRU <- x[5]
+   YRT <- x[6]
+   N <- X+YSU+YST+YRU+YRT
+   betaST <- params$alpha*params$betaSU
+   betaRU <- params$alpha2*params$betaSU
+   betaRT <- params$alpha3*betaRU
+   sigmaS <- params$FS*(params$nuSU+params$mu)/(1-params$FS)
+   sigmaR <- params$FR*(params$nuRU+params$mu+params$q)/(1-params$FR)
+   lambdaS <- as.numeric(lambdaS(params$betaSU,YSU,betaST,YST,params$pSU,YRU,params$pST,YRT,N))
+   lambdaR <- as.numeric(lambdaR(betaRU,YRU,betaRT,YRT,N))
+   dX <- params$pi - (params$c*(lambdaS+lambdaR)+params$mu)*X
+   dY <- (params$c*(lambdaS+lambdaR)+params$mu)*X
+   dYSU <- lambdaS*params$c*X + params$q*YRU + params$gS*YST - YSU*(sigmaS+params$nuSU+params$mu)
+   dYST <- YSU*sigmaS - YST*(params$gS + params$r + params$nuST + params$mu)
+   dYRU <- X*params$c*lambdaR + YRT*params$gR - YRU*(params$q + params$e*sigmaR + params$nuRU + params$
+   dYRT <- YRU*params$e*sigmaR + YST*params$r - YRT*(params$gR + params$nuRT + params$mu)
+   list(c(dX, dY, dYSU, dYST, dYRU, dYRT))
+ }
```

```
> require(deSolve)
> times <- seq(0, 20, by=1)  #solve for 20 years
```

```
> params <- list(betaSU=0.1, alpha=0.1+(0.5-0.1)/2 , pSU=0.5, pST=0.5, FS=0.7, FR=0.7, alpha2=0.5 , alph
> pop.size <- 800000*0.5*0.15
> xstart <- c(X=pop.size*0.7, Y=pop.size*0.3, YSU=pop.size*0.3*0.5*0.999,
+             YST=pop.size*0.3*0.5*0.85, YRU=pop.size*0.3*0.5*0.001, YRT=pop.size*0.3*0.5*0.15)
> out <- as.data.frame(lsoda(xstart, times, hiv.model.sf, params))
```

As in the example, we calculate the baseline number of infectious prevented.

```
> infections.prevented.baseline<-c()
> for(F in seq(0,0.98,by=0.02)){
+   params <- list(betaSU=0.1, alpha=0.1+(0.5-0.1)/2 , pSU=0.5, pST=0.5, FS=F, FR=F, alpha2=0.5 , alpha3
+   out <- as.data.frame(lsoda(xstart, times, hiv.model.sf, params))
+   new.infections.art <- sum(diff(out$Y))
+
+   params <- list(betaSU=0.1, alpha=0.1+(0.5-0.1)/2 , pSU=0.5, pST=0.5, FS=0, FR=0, alpha2=0.5 , alpha3
+   out <- as.data.frame(lsoda(xstart, times, hiv.model.sf, params))
+   new.infections.no.art <- sum(diff(out$Y))
+
+   infections.prevented.baseline <- c(infections.prevented.baseline, new.infections.no.art-new.infectio
+ }
```

Now we sample an 18-dimensional latin hypercube.

```
> require(lhs)            #add the lhs library
> h <- 500                #choose number of points
> lhs<-maximinLHS(h,18)   #simulate
```

Now we map to our parameter space.

```
> betaSU.min <- 0.0
> betaSU.max <- 0.2
> alpha.min <- 0.2
> alpha.max <- 0.4
> pSU.min <- 0.4
> pSU.max <- 0.6
> pST.min <- 0.4
> pST.max <- 0.6
> alpha2.min <- 0.4
> alpha2.max <- 0.6
> alpha3.min <- 0.4
> alpha3.max <- 0.6
> pi.min <- 2133-250
> pi.max <- 2133+250
> c.min <- 1.2
> c.max <- 2.2
> mu.min <- 0.0333-0.01
> mu.max <- 0.0333+0.01
> q.min <- 8.667-1
> q.max <- 8.667+1
> gS.min <- 0.04
```

```
> gS.max <- 0.06
> nuSU.min <- 0.08333-0.01
> nuSU.max <- 0.08333+0.01
> r.min <- 0.05
> r.max <- 0.15
> nuST.min <- 0.037-0.01
> nuST.max <- 0.037+0.01
> gR.min <- 0.05
> gR.max <- 0.15
> e.min <- 0.4
> e.max <- 0.6
> nuRU.min <- 0.037-0.01
> nuRU.max <- 0.037+0.01
> nuRT.min <- 0.037-0.01
> nuRT.max <- 0.037+0.01
> params.set <- cbind(
+   betaSU = lhs[,1]*(betaSU.max-betaSU.min)+betaSU.min,
+   alpha = lhs[,2]*(alpha.max-alpha.min)+alpha.min,
+   pSU = lhs[,3]*(pSU.max-pSU.min)+pSU.min,
+   pST = lhs[,4]*(pST.max-pST.min)+pST.min,
+   alpha2 = lhs[,5]*(alpha2.max-alpha2.min)+alpha2.min,
+   alpha3 = lhs[,6]*(alpha3.max-alpha3.min)+alpha3.min,
+   pi = lhs[,7]*(pi.max-pi.min)+pi.min,
+   c = lhs[,8]*(c.max-c.min)+c.min,
+   mu = lhs[,9]*(mu.max-mu.min)+mu.min,
+   q = lhs[,10]*(q.max-q.min)+q.min,
+   gS = lhs[,11]*(gS.max-gS.min)+gS.min,
+   nuSU = lhs[,12]*(nuSU.max-nuSU.min)+nuSU.min,
+   r = lhs[,13]*(r.max-r.min)+r.min,
+   nuST = lhs[,14]*(nuST.max-nuST.min)+nuST.min,
+   gR = lhs[,15]*(gR.max-gR.min)+gR.min,
+   e = lhs[,16]*(e.max-e.min)+e.min,
+   nuRU = lhs[,17]*(nuRU.max-nuRU.min)+nuRU.min,
+   nuRT = lhs[,18]*(nuRT.max-nuRT.min)+nuRT.min)
```

As in the example, we set 19 levels of F and use 250 points for evaluation.

```
> l <- 19
> h2 <-250
```

Finally we loop through our parameters.

```
> j <- 1
> data <- data.frame(matrix(rep(NA,l*h2*21),nrow=l*h2))
> for(i in 1:h2){
+   for (F in seq(0,0.9,length=l)){
+     xstart <- c(X=pop.size*0.7, Y=pop.size*0.3, YSU=pop.size*0.3*0.5*0.999,
+             YST=pop.size*0.3*0.5*0.85, YRU=pop.size*0.3*0.5*0.001, YRT=pop.size*0.3*0.5*0.15)
+
+     params <- as.list(c(params.set[i,], FS=0, FR=0))
+     out <- as.data.frame(lsoda(xstart, times, hiv.model.sf, params))
```

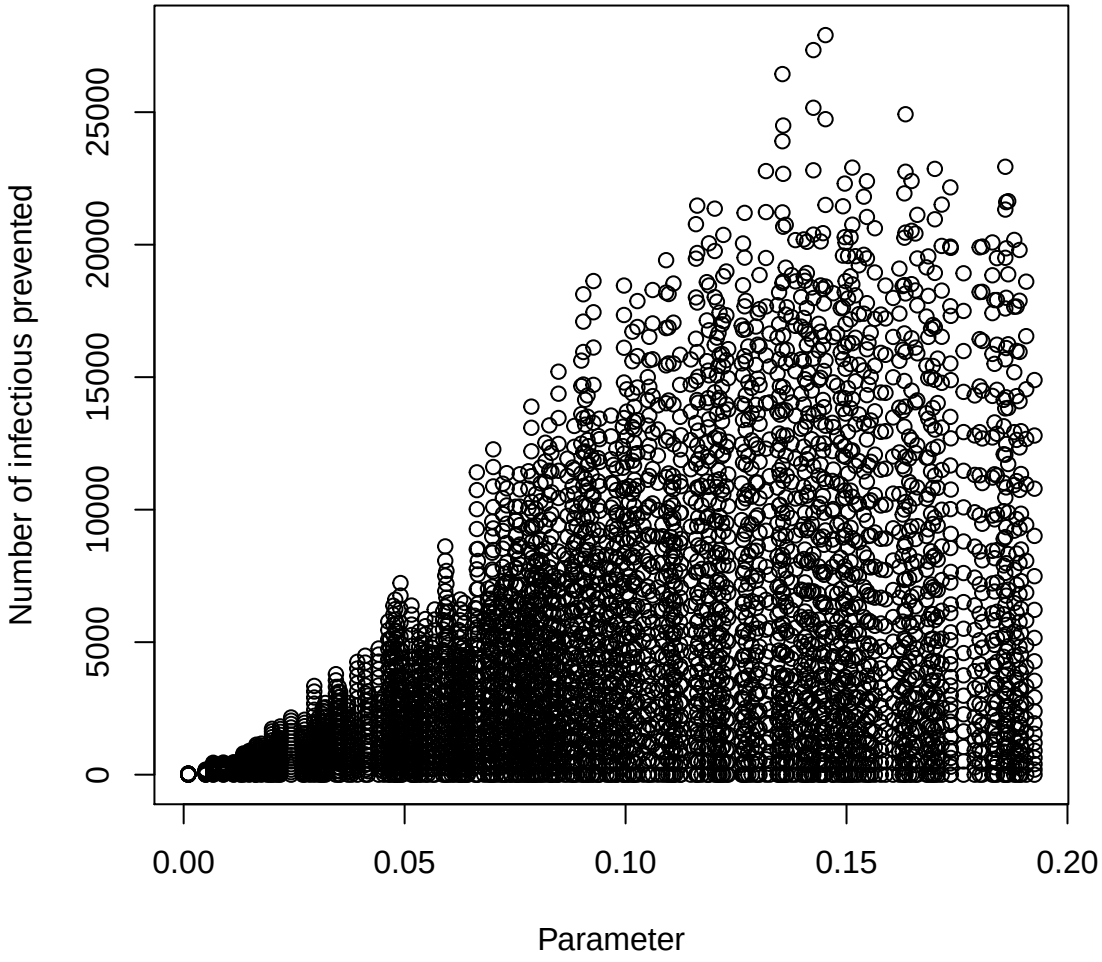
```
+     new.infections.no.art <- sum(diff(out$Y))
+
+     params <- as.list(c(params.set[i,], FS=F, FR=F))
+     out <- as.data.frame(lsoda(xstart, times, hiv.model.sf, params))
+     new.infections.art <- sum(diff(out$Y))
+
+     infections.prevented <- new.infections.no.art-new.infections.art
+     data[j,1:20] <- params
+     data[j,21] <- infections.prevented
+     j <- j+1
+   }
+ }
> names(data) <- c(names(params),'Infections.Prevented')
```

Now we write a simple loop. Inside the loop we calculate the correlation between the number of infections prevented and the random vector for each parameter obtained from the latin hypercube sample. If the correlation is higher than any previous, we store the result and retain the index of the variable (so we can go back and see which variable it was that was most correlated). Finally we plot the most correlated variable against number of infections to inspect visually.

```
> x<-1
> c.best <- 0
> for(i in 1:18){
+   c<-cor(data[,i],data[,21])
+   print(c)
+   if (abs(c)>c.best){
+     c.best <- abs(c)
+     x <- i
+   }
+ }
```

```
[1] 0.4340058
[1] -0.06231866
[1] -0.01347669
[1] -0.02972745
[1] -0.00729876
[1] 0.01803911
[1] 0.03875625
[1] 0.07116541
[1] -0.03918324
[1] -0.01410535
[1] 0.003416687
[1] -0.01311313
[1] -0.01520406
[1] -0.02065546
[1] 0.0001458761
[1] 0.02857944
[1] -0.00565512
[1] 0.02783349
```

```
> plot(data[,x], data[,21], xlab='Parameter', ylab='Number of infectious prevented')
```



Evidently, the first variable (β_{SU}) is most correlated with number of infections prevented. The correlation of this variable is 0.46.