Emissions Global Sensitivity Analysis: Draft methods

General methods:

Sample from empirically estimated distributions, separately for each EVT Group. Requires a baseline fuel bed for those fuel types not in the data base.

Use Iman method to induce empirical **rank** correlation structure among the variables. Claims that the method creates a rank correlation matrix of the input variables similar to the estimated one, while preserving the marginal distributions and the properties of the sampling scheme used to obtain the input vectors (so general and distribution-free):

Let K = # input variables, N = sample size; R = NxK (rows, columns) matrix with each column an independent permutations of an arbitrary set a(i), i = 1, . . . , N numbers (called "scores"). (*not sure yet what a(i) is*).

C* is the user-supplied rank correlation matrix, and let C = C. *P is the matrix such that PP' = C. P can be obtained by the Cholesky factorization scheme (?) used by Scheuer and Stoller (1962) (ref in Iman). Then R_iP' results in a vector that has the desired correlation matrix, and RP' = R* has the desired rank correlation matrix (or close to).

The example they give, the scores a(i) are the van der Waerden scores $\phi^{-1}(i/(N+1))$, i = 1,. . . ,N, in each column, where ϕ^{-1} is the inverse of the standard normal distribution. e.g., i = 1, then we need qnorm(1/(16)) when N = 15. The method then would be to randomly arrange the vector qnorm((1:N)/(N+1)) for each column. We can use "sample" for this

Their example:

```
# example rank correlation matrix, use cor(method=Spear,an)
C.star<-matrix(c(c(1,rep(0,5)),c(0,1,rep(0,4)),c(0,0,1,0,0,0),c(0,0,0,1,0.75,-0.70),
               c(0,0,0,0.75,1,-0.95),c(0,0,0,-.7,-.95,1)),ncol=6,byrow=TRUE)
P.trans<-chol(C.star) # this returns the cholesky factorization, P' in the paper
P.mat<-t(P.trans) # The transpose of the above matrix is the P matrix in the paper
N<-15
a.vals<-qnorm((1:N)/(N+1))
K<-6
R.mat<-matrix(NA,ncol=K,nrow=N)
for(j in 1:K)
{
  R.mat[,j]<-sample(a.vals)
}
R.star<-R.mat%*%P.trans
# so now we have a matrix with something like the correlation structure indicated by C*

mu.k<-c(1.5,0.6,2.8,3.5,1.8,0.1)
sigma.k<-c(0.4,0.1,0.9,0.5,0.25,0.8)
# Now generate desired sample structure, NxK, call X
# and arrange each column such that the rank of the observation
# in each row matches the rank in that row for R*. This
# preserves the RANK correlation structure
X.mat<-matrix(NA,ncol=K,nrow=N)
X.sort<-matrix(NA,ncol=K,nrow=N)
for(j in 1:K)
{
  X.mat[,j]<-rlnorm(N,mu.k[j],sigma.k[j])
```

```
  tmp.sort<-sort(R.star[,j],index.return=TRUE)
# so tmp.sort$ix gives the row number for each rank,
# i.e., tmp.sort$ix[i] gives the row number of the ith ordered statistic.
# inversely, which(tmp.sort$ix==i) gives the ordered statistic of the ith row
  tmp2.sort<-sort(X.mat[,j],index.return=TRUE)
# so now we replace the ith row in X with the value of the same rank as
# the ith row in R.star
  for(i in 1:N)
    X.sort[i,j]<-X.mat[tmp2.sort$ix[which(tmp.sort$ix==i)],j]
}
# Works in simple example!
# although note, especially with a small sample we get
# some substantial correlation in the zero-correlation
# pairwise relationships

round(cor(X.sort,method="spearman"),digits=3)
```

```
##        [,1]   [,2]   [,3]   [,4]   [,5]   [,6]
## [1,]  1.000  0.096  0.654 -0.111 -0.054  0.050
## [2,]  0.096  1.000  0.418 -0.275 -0.457  0.407
## [3,]  0.654  0.418  1.000 -0.307 -0.379  0.300
## [4,] -0.111 -0.275 -0.307  1.000  0.864 -0.889
## [5,] -0.054 -0.457 -0.379  0.864  1.000 -0.964
## [6,]  0.050  0.407  0.300 -0.889 -0.964  1.000
```

```
C.star
```

```
##      [,1] [,2] [,3]  [,4]  [,5]  [,6]
## [1,]    1    0    0  0.00  0.00  0.00
## [2,]    0    1    0  0.00  0.00  0.00
## [3,]    0    0    1  0.00  0.00  0.00
## [4,]    0    0    0  1.00  0.75 -0.70
## [5,]    0    0    0  0.75  1.00 -0.95
## [6,]    0    0    0 -0.70 -0.95  1.00
```

Also Iman proposes a variance reduction method that produces closer (more consistent) correlation matrices. Let T = the realized correlation matrix for R above. Use Cholesky factorization to find Q such that T = QQ' (as with C and P above). We can then solve for S = PQ^{-1}. Then we have R*_b=RS'. Not sure if we necessarily need this.

```
T.mat<-cor(R.mat,method="spearman")
Q.trans<-chol(T.mat)
Q.mat<-t(T.mat)
S.mat<-P.mat%*%solve(T.mat)
R.star.b<-R.mat%*%t(S.mat)
X.sort.b<-matrix(NA,ncol=K,nrow=N)
for(j in 1:K)
{
  tmp.sort<-sort(R.star.b[,j],index.return=TRUE)
# so tmp.sort$ix gives the row number for each rank,
# i.e., tmp.sort$ix[i] gives the row number of the ith ordered statistic.
# inversely, which(tmp.sort$ix==i) gives the ordered statistic of the ith row
  tmp2.sort<-sort(X.mat[,j],index.return=TRUE)
# so now we replace the ith row in X with the value of
# the same rank as the ith row in R.star
  for(i in 1:N)
```

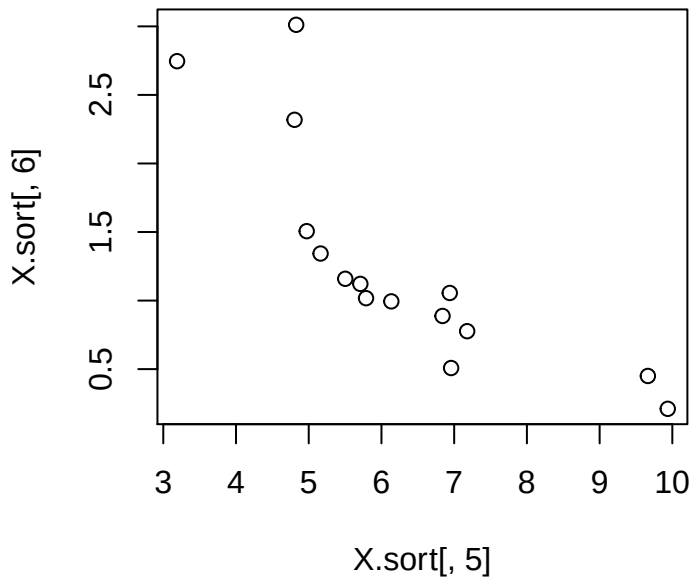
```
    X.sort.b[i,j]<-X.mat[tmp2.sort$ix[which(tmp.sort$ix==i)],j]
}
# Works in simple example!

round(cor(X.sort.b,method="spearman"),digits=3)
```

```
##        [,1]   [,2]   [,3]   [,4]   [,5]   [,6]
## [1,]  1.000  0.068 -0.632  0.189 -0.018  0.214
## [2,]  0.068  1.000 -0.193 -0.146  0.407 -0.411
## [3,] -0.632 -0.193  1.000  0.025  0.246 -0.329
## [4,]  0.189 -0.146  0.025  1.000  0.414 -0.254
## [5,] -0.018  0.407  0.246  0.414  1.000 -0.900
## [6,]  0.214 -0.411 -0.329 -0.254 -0.900  1.000
```

```
C.star
```

```
##      [,1] [,2] [,3]  [,4]  [,5]  [,6]
## [1,]    1    0    0  0.00  0.00  0.00
## [2,]    0    1    0  0.00  0.00  0.00
## [3,]    0    0    1  0.00  0.00  0.00
## [4,]    0    0    0  1.00  0.75 -0.70
## [5,]    0    0    0  0.75  1.00 -0.95
## [6,]    0    0    0 -0.70 -0.95  1.00
```

```
#win.graph(width=4,height = 4)
plot(X.sort[,5],X.sort[,6])
```



Now let's test this on an example EVT group. Some considerations–do we go with the full correlation matrix, or only those deemed "statistically" significant, and by what measure? Also, back to zeroes! We need to include them in the correlation consideration, and theoretically they shouldn't be a problem with a rank correlation v. a pearson correlation.