PE&RC Postgraduate course
UA&SV

Tuesday 10-12-2024: Uncertainty propagation in vegetation indices using Monte Carlo simulation

Introduction

In this lab we will use Monte Carlo simulation for analysing uncertainty propagation through the two vegetation indices Simple Ratio (SR) and Normalised Difference Vegetation Index (NDVI):

$$SR = NIR / RED$$

$$NDVI = (NIR - RED) / (NIR + RED)$$

and compare the results with those obtained using 1st order Taylor series approximation.

Sampling from RED and NIR

Monte Carlo (MC) simulation is a simple method in which the model (in this case a vegetation index) is applied on a sample of the uncertain inputs to assess uncertainty in the output. We thus need a sample from the random variables RED and NIR and feed that into the formulas listed in the introduction. Again we will use:

$$\sigma_{RED} = 0.025, \quad \sigma_{NIR} = 0.03, \quad \rho = 0.8.$$

If we assume a normal bivariate joint distribution for RED and NIR we can write:

$$X \sim \mathcal{N}_2(\mu, \Sigma),$$

where μ denotes the vector of means of RED and NIR and Σ is the 2×2 covariance matrix:

$$\mu = \begin{bmatrix} \mu_{RED} \\ \mu_{NIR} \end{bmatrix} \qquad \Sigma = \begin{bmatrix} \sigma_{RED}^2 & \rho\sigma_{RED}\sigma_{NIR} \\ \rho\sigma_{RED}\sigma_{NIR} & \sigma_{NIR}^2 \end{bmatrix}$$

A realisation from this bivariate normal distribution can be obtained by generating a vector Z of two independent variates from the standard normal distribution, multiplying this Z by a matrix M such that M multiplied by its transpose equals Σ, i.e., $M \cdot M^T = \Sigma$ and, next, adding the mean vector μ, i.e.,

$$X = M \cdot Z + \mu$$

(see e.g. http://en.wikipedia.org/wiki/Multivariate_normal_distribution#Drawing_values_from_the_distribution). It is the multivariate equivalent of computing realizations of a normally distributed random variable with mean μ and standard deviation σ from realizations of the standard normal distribution.

The matrix M can be found by Choleski factorization of the covariance matrix. The above is implemented in the following block of R code, which you can copy to a script file. Note that below the variable n is set to generate one thousand realisations of the vector *X*.

```
s_RED <- 0.025    # sigma RED
s_NIR <- 0.03     # sigma NIR
rho <- 0.8        # correlation errors in NIR and RED

# covariance matrix and Choleski factorization
covmat <- matrix(c(s_RED^2,rep(rho*s_RED*s_NIR,2),s_NIR^2),2,2)
M <- t(chol(covmat))
all(M %*% t(M) == covmat) # just a check; %*% is matrix multiplication

set.seed(1234)  # initialize pseudo random number generator
n <- 1000       # sample size

Z <- matrix(rnorm(2*n),2,n) # 2 rows, n columns; independent draws from
#                             the standard normal distribution
devs <- t(M %*% Z)  # independent draws from multivariate normal
                    # distribution with covmat
plot(devs, pch=19, cex=0.7, xlab="red error", ylab="NIR error")
```

Single point

It is convenient to write functions for the computation of the standard deviations of SR and NDVI, which you can apply on individual values for red and NDVI or series of data (cf. yesterday). Below, example code is given for the SR index. NDVI is left as an exercise.

```
# compute mean and sd SR from a series of realised deviations of
# red and nir (in x) and central values for red and nir
MC_SR <- function(red, nir, devs) {
  RED <- red+devs[,1]
  NIR <- nir+devs[,2]
  # ignore red reflectances < 0.025% (division by values close to zero)
  RED[RED < 0.00025] <- NA
  # return SR realizations without missing values
  na.omit(NIR/RED)
}

set.seed(1234567)  # initialize pseudo random number generator
n <- 200           # sample size
Z <- matrix(rnorm(2*n),2,n) # 2 rows, n columns; independent draws from
#                             standard normal distribution
devs <- t(M %*% Z)
SRsamp <- MC_SR(0.1, 0.6, devs)
sd(SRsamp)
mean(SRsamp)
```

Exercise 1: *Study the code on the previous page and add it to your script file.*

- *Display a histogram (using the function* `hist`*), displaying the distribution of the uncertain SR. Does he distribution look normal?*
- *Assess how large n should be to get stable estimates of σ_{SR} using red and NIR values of 0.1 and 0.6, respectively. You can do this by running the script several times with the same value for n without reseeding the pseudo random generator (see* `?rnorm`, `?Random`*). Next, repeat the procedure with a new value for n.*

Exercise 2: *Automate the above approach for a series of values for n, such that for each n, 50 samples of size n of the inputs are generated for each of which σ_{SR} is computed. Make and interpret a plot of the standard deviation of the variance of σ^2_{SR} (that is the square of σ_{SR}) for increasing values of n. The code in the box below gives a potential solution but the same can also be achieved using two nested for loops.*

```
N <- 1:100*1000  # a series of values for n
set.seed(123456) # set a seed for the PRN generator

sdSR <- function(n, red, nir) {
 Z <- matrix(rnorm(2*n),2,n)
 devs <- t(M %*% Z)
 sd(MC_SR(red, nir, devs))
}

funSR <- function(n, red, nir) replicate(50, sdSR(n, red, nir))

sdsSR <- sapply(N, function(n) funSR(n, 0.1, 0.6))

sd_var <- apply(sdsSR, 2, function(x) sd(x^2))

plot (sapply(N, rep, times=50), sdsSR, xlab="sample size",
 ylab="sd fom sample",
 main = "red = 0.1, NIR = 0.6", pch=16, cex=0.4)
plot (N, sd_var, xlab = "sample size", ylab="sd(var(SR))",
 main = "red = 0.1, NIR = 0.6")
lines(N, sd_var[100]/sqrt(N/N[100]), col="red") # inverse sqrt relation
```

- *Is the number of required runs affected when using greater (e.g. 0.6) values for the reflectance in the red band? Explain.*

Exercise 3: *Add functions for estimating μ_{NDVI} and σ_{NDVI} by MC. Only consult this code snippet if you don't know how to proceed.*

Exercise 4: *Repeat exercise 2 for NDVI to assess a suitable value for n.*

Contour plots

Yesterday, we used the `outer()` function for applying 1st order Taylor order functions on each combination of uncertain reflectances in the red and nir bands. For the Monte Carlo (MC) approach, it seems more convenient to first create a table (`data.frame`) with all those combined reflectances and next apply the MC on the rows of that table. The former is achieved with the function `expand.grid()`, while the latter can be achieved with `apply()`. The box on the next page provides some example code.

```
red <- 1:500/500   # generate a sequence 0.002, 0.004, ., 1.00
nir <- 1:500/500
rednir <- expand.grid(red, nir)

mu_sdSR <- function(x) {
  samp <- MC_SR(x[1], x[2], devs=devs)
  c(mean(samp), sd(samp))
}

n = 20000
set.seed(1234567)
Z <- matrix(rnorm(2*n),2,n)
devs <- t(M %*% Z)

mu_sd <- apply(rednir, 1, mu_sdSR)
MCmu_SR <- matrix(mu_sd[1,], 500, 500)
MCsd_SR <- matrix(mu_sd[2,], 500, 500)

CV_SR <- (MCsd_SR / MCmu_SR)
```

Exercise 5: *Make contour plots of the uncertainty about both SR and NDVI for RED and NIR ranging between 0.002 and 1.00, like you did yesterday using the 1st order Taylor series approximation. Use this code snippet only as a final resort*

- *Compare yesterday's plots with those of today. Where are they different and where are they similar? Reason about any differences between the plots.*

Image

Like yesterday, in his lab we disregard the effect of spatial correlation of NIR and RED errors within the image. Accounting for such correlation would be important if we want to assess uncertainty of the mean SR or NDVI over regions that have a different size than the image pixels. In such case, errors would partially cancel out if pixels within the aggregate are spatially independent while in the opposite case, they strengthen each other.

However, for now we will only consider the uncertainty in the vegetation indices at the original pixel support. In other words, we will not spatially aggregate uncertainties to larger spatial units, e.g. to assess uncertainty of the vegetation index averaged over agricultural fields. Tomorrow we *will* look into spatial and temporal correlation of errors, using a different case study.

Exercise 6: *Perform uncertainty analyses using the Monte Carlo method on the false_color image (FCLorraine.tif) using the functions of exercises 1 and 2 and compare the results with those obtained yesterday with the Taylor series approximation method. Here is a code snippet, should you get stuck.*

In contrast to the 1st order Taylor series method, Monte Carlo simulation also provides information about the output distribution in uncertainty propagation assessments. This information can be used for assessing the probability of exceeding (or remaining below) a critical threshold on the output variable.

Exercise 7: *Assess the probability of NDVI $<$ 0.3 (a typical value for differentiating between vegetated and non-vegetated areas) and interpret the result. Reason about a way for computing such probability using the 1st order Taylor series method. If time allows you can also try to implement it.*