Spatial uncertainty propagation analysis

Case study with spatially variable standard deviation - slope calculations with a digital elevation model (DEM)

Kasia Sawicka and Gerard Heuvelink

2017-02-12

### Introduction/Problem definition

In many geographical studies a DEM is a critical variable, because DEM-derived variables, such as slope, aspect, curvature and viewshed are of great importance in many types of analysis. However, a DEM is only an approximation of the real elevation in the area. It contains errors. Insight into DEM error (uncertainty) propagation through the calculation of, for example, slope, is therefore crucial. We can use the Monte Carlo (MC) method to analyse how the error propagates through spatial operations and models. This method is briefly described below.

The MC method is fairly straightforward in application, but in case of spatially distributed variables like elevation one should consider taking spatial autocorrelation into account. That is because the model output uncertainty may be influenced by the spatial correlation in the input. For example, slope calculations are quite sensitive to the degree of spatial autocorrelation in DEM uncertainty (Heuvelink, 1998).

### Monte Carlo methodology for spatial uncertainty analysis with spatially variable sd

The uncertainty propagation analysis approach applied here is based on the Monte Carlo method that computes the output of the model repeatedly, with input values that are randomly sampled from their marginal or joint probability distribution function (pdf). The set of model outputs forms a random sample from the output pdf, so that parameters of the distribution, such as the mean, variance and quantiles, can be estimated from the sample. The method thus consists of the following steps:

1. Characterise uncertain model inputs with (spatial) pdfs.
2. Repeatedly sample from (spatial) pdfs of uncertain inputs.
3. Run model with sampled inputs and store model outputs.
4. Compute summary statistics of model outputs.

Note that the above ignores uncertainty in model parameters and model structure, but these can easily be included if available as pdfs. A random sample from the model inputs can be obtained using an appropriate pseudo-random number generator.

For uncertain spatially distributed continuous variables, such as elevation, we assume the following geostatistical model:

Z(x)= μ(x)+ σ(x)∙ε(x)

where x is geographic location, μ is the (deterministic) mean of Z, σ is its standard deviation and ε is a standard normal (hence zero mean and unit variance), second-order stationary stochastic residual, whose spatial autocorrelation is modelled with a semivariogram or correlogram. Both μ and σ may vary in space so that spatial trends and spatially variable uncertainty can be taken into account. In the case of elevation, it makes sense to let μ be equal to the DEM while σ may have greater values in mountainous areas than in flat terrain (e.g. Beekhuizen, et al., 2011). The random sample is drawn from the pdf of ε to further calculate a sample from Z.

### DEM uncertainty propagation analysis with 'spup'

#### Preliminaries - load and view the data

The example data for slope calculations are a 30m resolution mean DEM and standard deviation from the Zlatibor region in Serbia.

The 'DEM' data contains two spatial objects: a mean and the standard deviation map. It also has a saved function that calculates slope from elevation that will be used later.

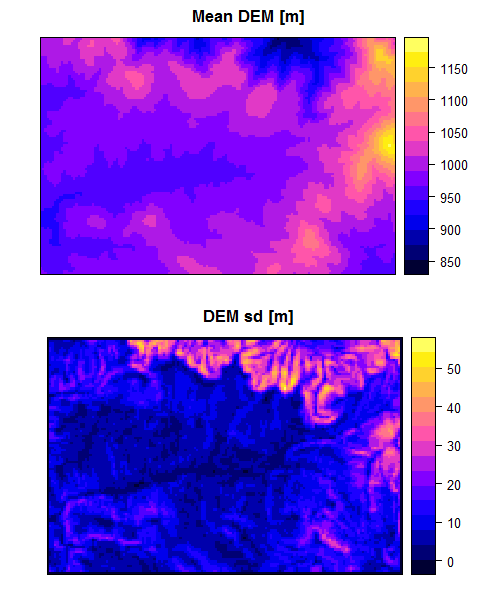
# load packages  
library(sp)  
library(spup)  
library(gridExtra)  
  
# load and view the data  
data(DEM)  
str(dem30m)

Formal class 'SpatialGridDataFrame' [package "sp"] with 4 slots  
 ..@ data :'data.frame': 15000 obs. of 1 variable:  
 .. ..$ Elevation: num [1:15000] 1010 1010 1008 1007 1005 ...  
 ..@ grid :Formal class 'GridTopology' [package "sp"] with 3 slots  
 .. .. ..@ cellcentre.offset: Named num [1:2] 7394264 4842014  
 .. .. .. ..- attr(\*, "names")= chr [1:2] "x" "y"  
 .. .. ..@ cellsize : num [1:2] 30 30  
 .. .. ..@ cells.dim : int [1:2] 150 100  
 ..@ bbox : num [1:2, 1:2] 7394249 4841999 7398749 4844999  
 .. ..- attr(\*, "dimnames")=List of 2  
 .. .. ..$ : chr [1:2] "x" "y"  
 .. .. ..$ : chr [1:2] "min" "max"  
 ..@ proj4string:Formal class 'CRS' [package "sp"] with 1 slot  
 .. .. ..@ projargs: chr "+init=epsg:3857 +proj=merc +a=6378137 +b=6378137 +lat\_ts=0.0 +lon\_0=0.0 +x\_0=0.0 +y\_0=0 +k=1.0 +units=m +nadgrids=@null +no\_def"| \_\_truncated\_\_

str(dem30m\_sd)

Formal class 'SpatialGridDataFrame' [package "sp"] with 4 slots  
 ..@ data :'data.frame': 15000 obs. of 1 variable:  
 .. ..$ Elevation\_sd: num [1:15000] 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 ...  
 ..@ grid :Formal class 'GridTopology' [package "sp"] with 3 slots  
 .. .. ..@ cellcentre.offset: Named num [1:2] 7394264 4842014  
 .. .. .. ..- attr(\*, "names")= chr [1:2] "x" "y"  
 .. .. ..@ cellsize : num [1:2] 30 30  
 .. .. ..@ cells.dim : int [1:2] 150 100  
 ..@ bbox : num [1:2, 1:2] 7394249 4841999 7398749 4844999  
 .. ..- attr(\*, "dimnames")=List of 2  
 .. .. ..$ : chr [1:2] "s1" "s2"  
 .. .. ..$ : chr [1:2] "min" "max"  
 ..@ proj4string:Formal class 'CRS' [package "sp"] with 1 slot  
 .. .. ..@ projargs: chr "+init=epsg:3857 +proj=merc +a=6378137 +b=6378137 +lat\_ts=0.0 +lon\_0=0.0 +x\_0=0.0 +y\_0=0 +k=1.0 +units=m +nadgrids=@null +no\_def"| \_\_truncated\_\_

grid.arrange(spplot(dem30m, main = list(label = "Mean DEM [m]", cex = 1)),  
 spplot(dem30m\_sd, main = list(label = "DEM sd [m]", cex = 1)))



#### Define uncertainty model (UM) for elevation

The first step in uncertainty propagation analysis is to define an uncertainty model for the uncertain input variable, here elevation, that will be used in the Monte Carlo uncertainty propagation analysis.

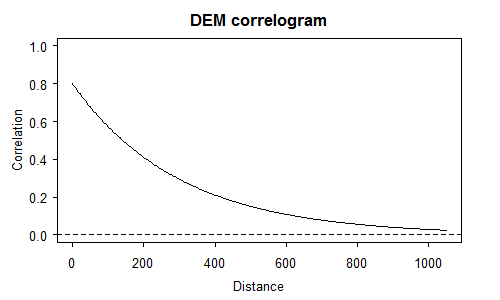
In case of elevation, the ε(x) are spatially correlated and in order to include this in the analysis, we need to describe it by spatial correlogram parameters. The makecrm() function collates all necessary information into a list.

Let us assume that the spatial autocorrelation of the DEM errors is an exponentially decreasing function with a short-distance correlation of 0.8 and a range parameter of 300m.

# define spatial correlogram model  
dem\_crm <- makecrm(acf0 = 0.8, range = 300, model = "Exp")

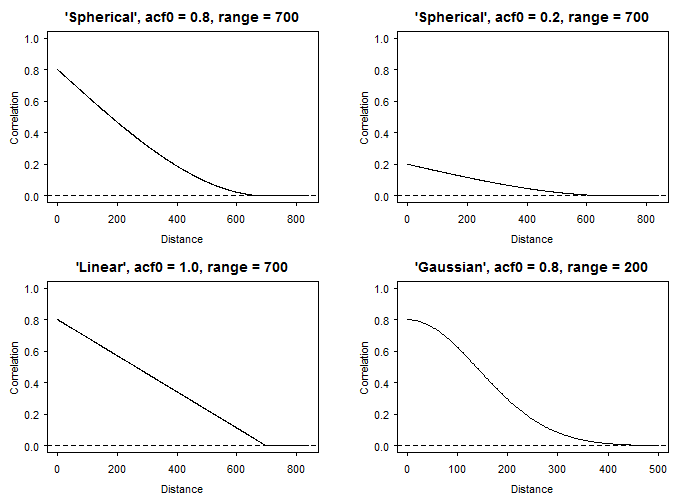
We can view the correlogram by plotting it.

plot(dem\_crm, main = "DEM correlogram")



Spatial correlograms summarise patterns of spatial autocorrelation in data and model residuals. They show the degree of correlation between values at two locations as a function of the separation distance between the locations. In the case above the correlation declines with distance, as is usually the case. The correlation becomes negligibly small for distances greater than 800 m. Notice also that the correlation is not perfect at distances close to zero. This signifies the so-called ‘nugget’ effect. The nugget effect, shape of the correlation function and maximum range can be modified by changing the parameters of the makecrm() function. Try, for example, these combinations and see how they look like:

par(mfrow = c(2, 2))  
crm <- makecrm(acf0 = 0.8, range = 700, model = "Sph")   
plot(crm, main = "'Spherical', acf0 = 0.8, range = 700")  
crm <- makecrm(acf0 = 0.2, range = 700, model = "Sph")   
plot(crm, main = "'Spherical', acf0 = 0.2, range = 700")  
crm <- makecrm(acf0 = 0.8, range = 700, model = "Lin")   
plot(crm, main = "'Linear', acf0 = 1.0, range = 700")  
crm <- makecrm(acf0 = 0.8, range = 200, model = "Gau")   
plot(crm, main = "'Gaussian', acf0 = 0.8, range = 200")



In order to complete the description of the uncertain variable we use the defineUM() function that collates all information about the DEM uncertainty into one object. The minimum information required is:

* logical value that indicates if the object is uncertain.
* distribution type to sample from. In case of variables with spatially correlated errors only the normal distribution is supported. For details on supported distributions and required parameters see ?defineUM.
* list of distribution parameters, for example a mean and a standard deviation (sd) for the normal distribution. In the case presented here, these are maps of the mean DEM and standard deviation of the DEM error.
* correlogram model.

# define uncertainty model for the DEM  
demUM <- defineUM(uncertain = TRUE, distribution = "norm",   
 distr\_param = c(dem30m, dem30m\_sd), crm = dem\_crm)  
class(demUM)

[1] "MarginalNumericSpatial"

#### Generate possible realities of DEM

Generating possible realities of the elevation can be completed by using the genSample() function. The required information to pass to the function includes:

* uncertain object (as defined above).
* number of realizations to return.
* sampling method. In case of spatially correlated variables, the method "ugs" (method based on unconditional Gaussian simulation) is recommended, otherwise spatial correlation will not be taken into account. Other sampling methods include "randomSampling" and "stratifiedSampling". See ?genSample for more details.

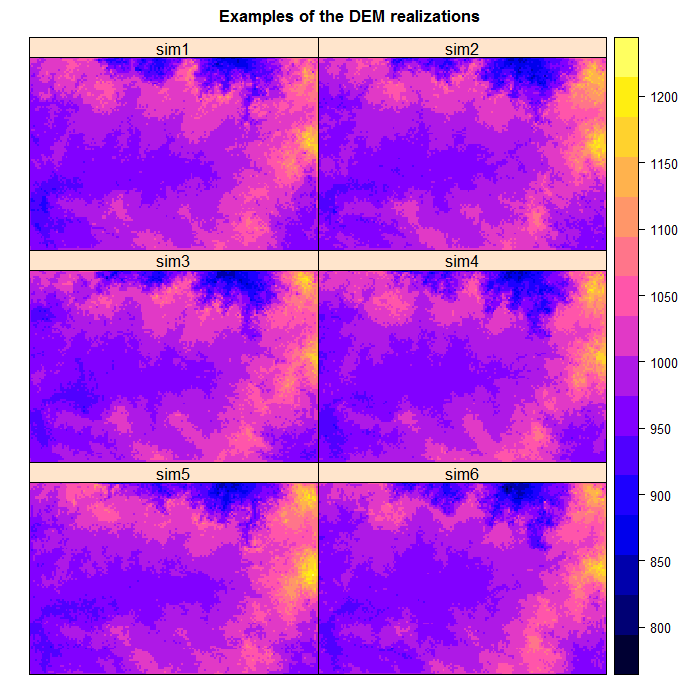
Additional parameters may be also specified. For example, sampling of spatially correlated variable is based on the 'gstat' package that allows for limiting the number of nearest observations that should be used for simulation.

The sample must be large to obtain stable results. Let us run the sampling to obtain 100 realizations. Note that the argument 'asList' has been set to FALSE. This indicates that the sampling function will return an object of the same class as maps of the elevation mean and sd. This is useful if you want to visualize the sample or compute summary statistics quickly.

# create realizations of the DEM  
dem\_sample <- genSample(UMobject = demUM, n = 100, samplemethod = "ugs", nmax = 20, asList = FALSE)

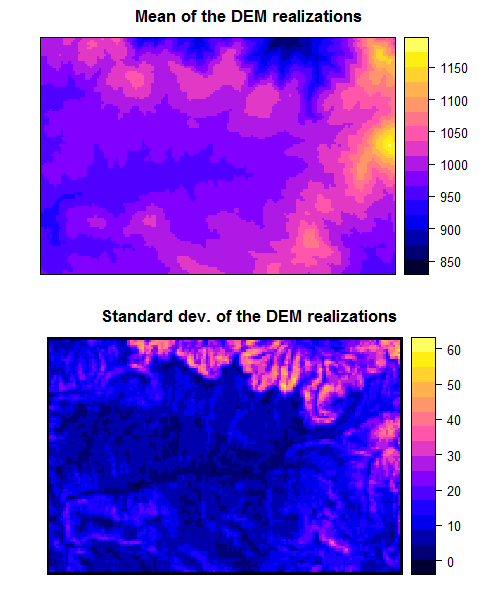
[using unconditional Gaussian simulation]

# view several realizations of DEM  
spplot(dem\_sample[c(5,6,3,4,1,2)], main = list(label = "Examples of the DEM realizations", cex = 1))



We can view the mean and standard deviation of the sampled elevation. If the sample size was very large then the sample mean would be close to 'dem30m' and the sd close to 'dem30m\_sd':

# compute and plot the slope sample statistics  
# e.g. mean and standard deviation  
dem\_sample\_mean <- mean(dem\_sample)  
dem\_sample\_sd <- sd(dem\_sample)  
grid.arrange(spplot(dem\_sample\_mean, main = list(label = "Mean of the DEM realizations", cex = 1)),  
 spplot(dem\_sample\_sd, main = list(label = "Standard dev. of the DEM realizations", cex = 1)))

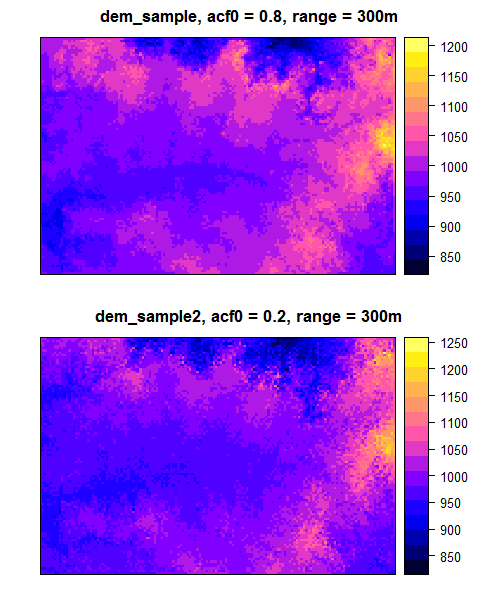


The spatial pattern of the generated realizations depends on the degree of spatial auto-correlation. For instance, notice that the realizations become more ‘noisy’ if we assume less spatial-autocorrelation:

dem\_crm2 <- makecrm(acf0 = 0.2, range = 300, model = "Exp")  
demUM2 <- defineUM(uncertain = TRUE, distribution = "norm", distr\_param = c(dem30m, dem30m\_sd), crm = dem\_crm2)  
dem\_sample2 <- genSample(UMobject = demUM2, n = 100, samplemethod = "ugs", nmax = 20, asList = FALSE)

[using unconditional Gaussian simulation]

grid.arrange(spplot(dem\_sample, c(1), main = list(label = "dem\_sample, acf0 = 0.8, range = 300m", cex = 1)),  
 spplot(dem\_sample2, c(1), main = list(label = "dem\_sample2, acf0 = 0.2, range = 300m", cex = 1)))



Can you spot the difference? Higher auto-correlation yields ‘smoothened’ realizations. Lower values produce a more ‘noisy’ field.

#### Uncertainty propagation through the model that calculates slope using elevation as input

In order to perform uncertainty propagation analysis using 'spup', the model through which uncertainty is propagated needs to be defined as an R function. The 'DEM' data object includes an example of a pre-defined model that calculates slope using elevation as input. In this case the function is based on the terrain() function from the **raster** package.

# view the model  
Slope

function(DEM, ...) {  
 require(raster)  
 demraster <-   
 DEM %>%  
 raster()  
   
 demraster %>%  
 terrain(opt = 'slope', ...) %>%  
 as("SpatialGridDataFrame")  
}

The propagation of uncertainty occurs when the model is run with an uncertain input. Running the model with a sample of realizations of uncertain input variable(s) yields an equally large sample of model outputs that can be further analysed. To run the Slope model with the elevations realizations we use the propagate() function. The propagate() function takes as arguments:

* a sample from the uncertain model inputs and any other remaining model inputs and parameters as a list.
* the model as a function in R.
* the number of Monte Carlo runs. This can be equal or smaller than the number of realizations of the uncertain input variable(s).

In order to run the propagation function it is necessary to save the sample of an uncertain input variable in a list. We can either coerce the existing 'dem\_sample' object or get it automatically by setting up the 'asList' argument of genSample() to TRUE.

# coerce SpatialGridDataFrame to a list of individual SpatialGridDataFrames  
dem\_sample <- map(1:ncol(dem\_sample), function(x){dem\_sample[x]})  
  
# or sample from uncertain input and save it in a list  
dem\_sample <- genSample(UMobject = demUM, n = 100, samplemethod = "ugs", nmax = 20, asList = TRUE)

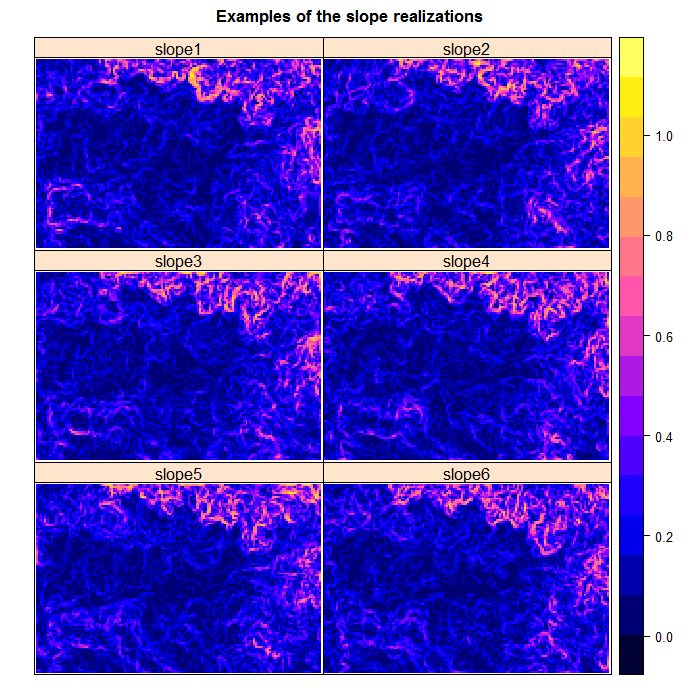
[using unconditional Gaussian simulation]

# run uncertainty propagation  
slope\_sample <- propagate(dem\_sample, model = Slope, n = 100)

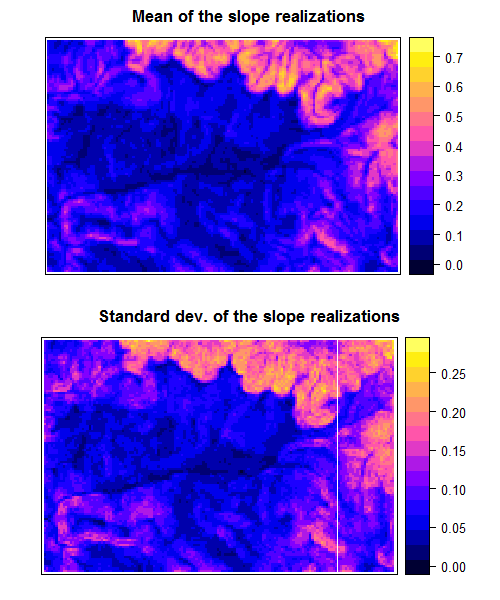
#### Visualization of results

We can now view the sample of model output realizations (i.e. slope) and visualize uncertainty by calculating and plotting the sample mean and standard deviation. In our case we need to coerce the output of the propagationfunction saved as a list back to a SpatialGridDataFrame.

# coerce slopes list to a SpatialGridDataFrame  
s <- slope\_sample[[1]]  
for (i in 2:length(slope\_sample)) {  
 s@data[i] <- slope\_sample[[i]]@data  
}  
names(s@data) <- paste("slope", c(1:ncol(s)), sep = "")  
slope\_sample <- s  
rm(s)  
  
# view the sample of the model output  
spplot(slope\_sample[c(5,6,3,4,1,2)], main = list(label = "Examples of the slope realizations", cex = 1))

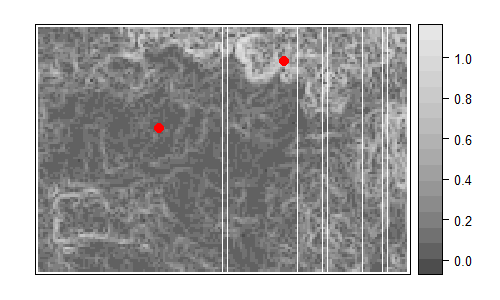


# compute and plot slope sample statistics  
# e.g. mean and standard deviation  
slope\_mean <- mean(slope\_sample)  
slope\_sd <- sd(slope\_sample, na.rm = TRUE) # na.rm = TRUE is necessary, because slope cannot be calculated at the border  
grid.arrange(spplot(slope\_mean, main = list(label = "Mean of the slope realizations", cex = 1)),  
 spplot(slope\_sd, main = list(label = "Standard dev. of the slope realizations", cex = 1)))

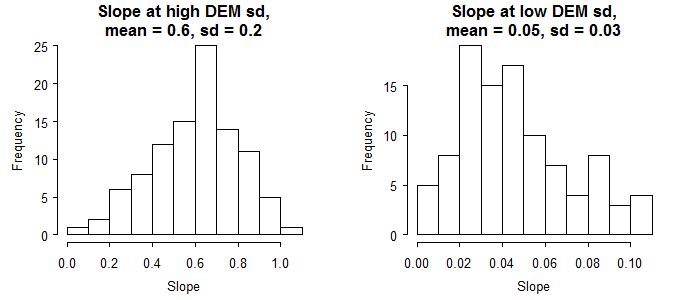


We can view example of slope realizations at locations:

# selec a couple of locations and plot points on the slope map  
loc1 <- 2200  
loc2 <- 6200  
points <- data.frame(t(coordinates(slope\_sample)[loc1,]))  
points[2,] <- t(coordinates(slope\_sample)[loc2,])  
coordinates(points) <- ~ s1 + s2  
proj4string(points) <- proj4string(slope\_sample)  
spplot(slope\_sample, c(1), col.regions = grey.colors(16),  
 sp.layout = c('sp.points', points, col = "red", pch = 19, cex = 1.2))



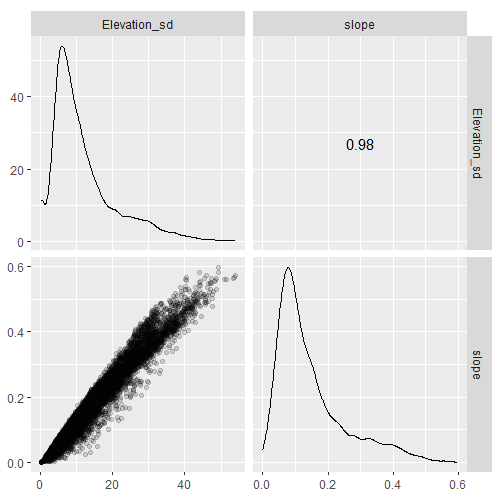
l\_mean <- mean(as.numeric(slope\_sample@data[loc1,]))  
l\_sd <- sd(as.numeric(slope\_sample@data[loc1,]))  
h\_mean <- mean(as.numeric(slope\_sample@data[loc2,]))  
h\_sd <- sd(as.numeric(slope\_sample@data[loc2,]))  
  
par(mfrow = c(1,2))  
hist(as.numeric(slope\_sample@data[loc1,]), main = paste("Slope at high DEM sd,", "\n",  
 "mean = ", round(l\_mean, 2), ", sd = ", round(l\_sd, 2), sep = ""), xlab = "Slope")  
hist(as.numeric(slope\_sample@data[loc2,]), main = paste("Slope at low DEM sd,", "\n",  
 "mean = ", round(h\_mean, 2), ", sd = ", round(h\_sd, 2), sep = ""), xlab = "Slope")



Since slope is a nearly linear combination of elevation differences, the slope sd has a similar spatial pattern as elevation sd. Therefore at locations with high elevation sd we get much higher uncertainty in slope predictions.

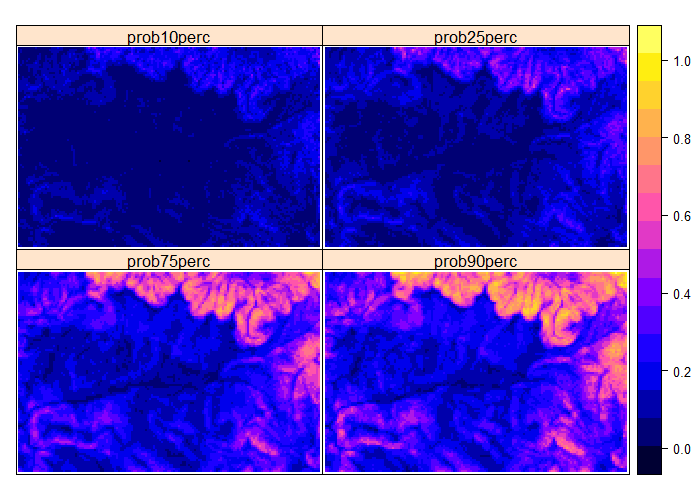
# scatter plot of slope against elevation  
library(GGally)  
slope1 <- Slope(dem30m)  
names(slope1@grid@ cellcentre.offset) <- c("x", "y")  
df <- cbind(dem30m\_sd@data, slope1@data)  
ggscatmat(data = df, alpha=0.15)

Warning: Removed 496 rows containing non-finite values (stat\_density).



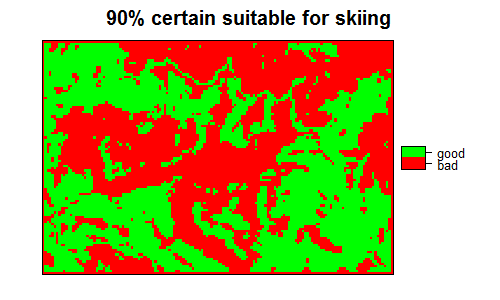
We can also look at specific quantiles of the slope sample.

# or quantiles  
slope\_q <- quantile(slope\_sample, probs = c(0.1, 0.25, 0.75, 0.9), na.rm = TRUE)  
spplot(slope\_q[c(3,4,1,2)], mail = list(label = "Quantiles of slope realizations", cex = 1))



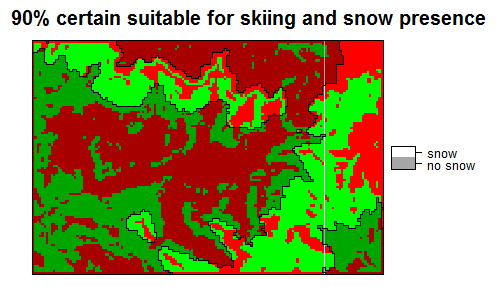
For example, let us identify locations with slopes that are possible (slope > 0.2) safe (slope < 0.5) for tourist skiing with probability 90%.

# identify areas of slope > 0.2 and < 0.5  
slope\_q$safep90 <- NA  
slope\_q$safep90[slope\_q$prob90perc < 0.5 & slope\_q$prob90perc > 0.2] <- "good"  
slope\_q$safep90[is.na(slope\_q$safep90)] <- "bad"  
slope\_q$safep90 <- as.factor(slope\_q$safep90)  
spplot(slope\_q, "safep90", col.regions = c("red","green"), main = "90% certain suitable for skiing")



Additionally to guarantee snow we can select areas with elevation > 1000m.

# identify areas of elevation > 1000m and overlay with the slope map above  
dem30m$Elev1000 <- NA  
dem30m$Elev1000[dem30m$Elevation > 1000] <- "snow"  
dem30m$Elev1000[is.na(dem30m$Elev1000)] <- "no snow"  
dem30m$Elev1000 <- as.factor(dem30m$Elev1000)  
  
snow <- dem30m[2]  
snow\_poly <- as(as(snow, "SpatialPixelsDataFrame"), "SpatialPolygonsDataFrame")  
snow\_poly <- aggregate(snow\_poly, by = "Elev1000")  
  
transparent\_grey <- rgb(0,0,0, alpha = 0.35)  
spl <- list("sp.grid", slope\_q[5], col = c("red","green"))  
spplot(snow\_poly, col.regions = c(transparent\_grey, "transparent"),   
 main = "90% certain suitable for skiing and snow presence",  
 sp.layout = spl)



### Acknowledgements

The Zlatibor dataset was kindly provided by Prof. Branislav Bajat from the University of Belgrade, Serbia.

This project has received funding from the European Union’s Seventh Framework Programme for research, technological development and demonstration under grant agreement no 607000.

### References

Beekhuizen, J. G.B.M. Heuvelink, J. Biesemans and I. Reusen (2011), Effect of DEM uncertainty on the positional accuracy of airborne imagery. IEEE Transactions on Geoscience and Remote Sensing 49, 1567 1577.