ES7023 - Assignment 7: Uncertainty Quantification, Bootstrap and Spatial Prediction

David Lallemant

All models are uncertain, and it is imperative for modelers to examine and be transparent about the uncertainties in their models. Uncertainty is often classified into two categories:

1. Aleatory uncertainty:

Alleatory uncertainty refers to inherent uncertainty and randomness of natural processes. The outcome of a dice-roll is an example of aleatory uncertainty. This type of uncertainty is *irreducible*, in that there will always be variability in the underlying process. The term aleatory is derived from Latin *alea* refering to a game of chance, and in fact the French translation for "hazard" is "aléa".

2. Epistemic uncertainty

Epistemic uncertainty refers to uncertainty that stems from lack of knowledge: lack of understanding of underlying processes, imprecise measurements, etc. This type of uncertainty is *reducible*, through the collection of more data, more tests, more modeling, etc.

While these are common classifications of uncertainty, modelers often get caught in semantic discussions about whether a process is aleatory or epistemic. As an example, some may (and often do) argue that the uncertainty of a dice roll reflects lack of knowledge of the exact underlying physics and environmental state; if we knew them, we could predict the outcome of each roll. These are interesting debates, but not terribly helpful.

I include this discussion of aleatory and epistemic uncertainty so that you are familiar with these common terms, but more directly useful to us is to characterise uncertainty from their source:

1. **Input uncertainty:** variability in input variables (e.g. natural variability, errors in measurements, etc).
2. **Parameter uncertainty:** uncertainty in the model parameters / coefficients.
3. **Structural uncertainty:** uncertainty stemming from lack of knowledge of the underlying process. It refers to how accurately (or innacurately) a mathematical model describes the true (real-world) process. It is sometimes refered to as "model inadequacy" or "model bias."
4. **Others:** we will not go through an exhaustive list of sources of uncertainty, but these could include interpolation uncertainty, experimental uncertainty and others.

Ultimately, the final prediction uncertainty in a model is the result of the combination of numerous sources of uncertainty. We will explore bootstrap method and Monte Carlo uncertainty propagation methods to quantify prediction uncertainty in our landslide model.

As before, the data consists of the following parameters:

- `x` & `y` : coordinates
- `lslpts` : True (landslide occured), False (no landslide). This is the variable we will try to model and predict.
- `slope`: slope angle (°).
- `cplan`: plan curvature (rad m^{-1}) expressing the convergence or divergence of a slope and this water flow.

- cprof: profile curvature (rad m^{-1}) as a measure of flow acceleration, also know as downslope chane in slope angle.
- elev: elevation (m above sea level) as the representation of different altitudinal zones of vegetation and precipitation in the study area.
- log10_carea: the decadic logarithm of the catchment area (log10 m2) representing the amount of water flowing towards a location.

Read in the following data files:

```
setwd("/Users/david/Dropbox/My Mac (ASE-Mac2141)/Documents/R programing/")
landslide.train=read.csv("landslide_training_data.csv",header = T)

#load the raster stack. This will load up a raster stack called ta
load(file = "landslide_raster_stack")
```

Problem 1: Quick Modeling

- Fit a logistic model to predict landslides using the training dataset (use all predictor variables, except site coordinates). Display the fitted coefficients.
- We are interested in a specific site with the characteristics described below. Predict the logistic response (susceptibility of landslide) for this new site.
- Would you predict a landslide or no landslide at the new site (using a threshold of 0.5)?

```
newdata=data.frame(slope=50.15,cplan=0.0028,cprof=0.008, elev=2000,log10_carea=3.202)
```

Problem 2: Quantifying parameter uncertainty with the bootstrap method

The bootstrap method takes repeated samples of the dataset (with replacement) to estimate sampling errors on estimates of coefficients. The following code can be used to generate bootstrap samples:

```
bootsample<-landslide.train[sample(1:nrow(landslide.train),size = nrow(landslide.train),replace = T),]
```

Parameter uncertainty can then be obtained by itteratively fitting a model to a new bootstrap sample numerous times. The joint distribution of fitted coefficients generated from the bootstrap samples is a measure of model parameter uncertainty.

2.a. Quantifying parameter uncertainty

- Generate 1000 bootstrap samples from the training data and compute fitted coefficients for each sample (using the same model structure as in problem 1.a).
- Compute the mean and standard deviation of coefficients.
- How do the fitted coefficients from problem 1 compare to the mean values of the bootstrap coefficients calculated in the previous step?
- Show in as few plots as possible (preferably a single plot) the distribution of coefficients and their correlation.
- Create a scatterplot of the joint distribution of the coefficients corresponding to slope and log10_carea. Add contours to better show the joint density distribution (hint: look up geom_density_2d() in the ggplot2 package). Comment on the results.

2.b. Prediction uncertainty due to parameter uncertainty

- For each of the bootstrap samples, compute the susceptibility of landslide for the new data point from problem 1.
- Make a histogram or density plot of the corresponding predicted susceptibility of landslide. Add to your plot a vertical line at the predicted response computed in problem 1. Discuss these results. How confident are you in your prediction of landslide/no-landslide at this new site?

Problem 3: Uncertainty propagation combining input and parameter uncertainties

We now find out that the remote-sensing based measurements at our site of interest have some uncertainties. From past calibration, we know that the measurement errors can be described by the following distributions:

- $slope_{measured} = slope_{true} + \epsilon_1$ where $\epsilon_1 \sim N(0, 5)$
- $cplan_{measured} = cplan_{true} + \epsilon_2$ where $\epsilon_2 \sim N(0, 0.01)$
- $cprof_{measured} = cprof_{true} + \epsilon_3$ where $\epsilon_3 \sim N(0, 0.005)$
- $elev_{measured} = elev_{true} + \epsilon_4$ where $\epsilon_4 \sim N(0, 0)$ (no uncertainty)
- $log10carea_{measured} = log10carea_{true} + \epsilon_5$ where $\epsilon_5 \sim N(0, 0.25)$

In other words, the measurement errors follow a Normal (Gaussian) distribution, are unbiased (mean = zero), and with standard deviation of 5, 0.01, 0.005, 0 and 0.25 respectively. Note: We assume that the measurements at our new site are uncertain, but the training data have no measurement uncertainty (this may not always be a fair assumption, and would add to further uncertainty in model coefficients).

We can now use Monte Carlo simulation to combine and propagate input variable uncertainty (measurement errors above) and parameter uncertainty (from problem 2) to get the full distribution of prediction uncertainty.

There are many ways to do this. The most straight-forward is the following:

Step 1. Generate a vector of normally distributed variables corresponding to $\epsilon_1, \epsilon_2, \epsilon_3, \epsilon_4, \epsilon_5$.

Step 2. Add this vector to the measured values at our site of interest. You now have a random realization of the input variables.

Step 3. Generate a bootstrap sample from the training set.

Step 4. Fit a model to the bootstrap sample.

Step 5. Predict the response using the bootstrap model (from step 4) on the random realization of input variables (from step 2).

Step 6. Repeat steps 1-6 numerous times (>1000).

Problem task:

- Conduct Monte Carlo uncertainty propagation combining input variable uncertainty and parameter uncertainty using the method described above, or any other method. Remember that the simulation needs to account for distribution of input variable errors and the *joint-distribution* of model coefficients.
- Make a histogram or density plot of the corresponding predicted response (susceptibility of landslide) accounting for uncertainty in both input variables and model coefficients.
- Add to your plot the vertical line at the predicted response computed in problem 1.
- Add to your plot the distribution of predicted response accounting only for uncertainty in model parameters (the distribution from problem 2.b.)
- Discuss the results. How confident are you in your prediction of landslide/no-landslide at this new site?

Problem 4: Spatial Prediction and Mapping

We've provided you a raster stack containing 5 raster layers, each corresponding to a predictor variable.

4.a. Computation of terrain variables

- Install the `raster` package (`install.packages("raster")`) and load the library.
- Create a new raster layer of aspect ratio using the `terrain()` function and the elevevation data, as shown below

```
aspect_ratio=terrain(ta$elev, opt = "aspect")
```

- Create a new raster layer of hillshade using the `hillShade` function, the slope data and the aspect ratio computed earlier (note that you will have to convert the slope into radians): $Slope_{rad} = Slope_{degrees} * \pi/180$
- Display a map of hillshade in greyscale (hillshade is purely for visualization)

4.b. Spatial prediction

- Using the rasterstack data provided along with your landslide prediction model, create a new raster layer of landslide susceptibility. Note: this can be done very simply using the spatial model prediction function `predict()` from the `raster` package.
- Create a map of landslide susceptibility.
- Overlay the landslide susceptibility layer onto the hillshade map.

4.c. Decision making

A developer wants to build a new condominium at the following coordinate.

```
dev_coordinate=data.frame(x=714717.7, y=9560497)
```

- Use the `extract` function from the `raster` package to extract the values of the predictor variables from the raster-stack at the coordinates of interest. (Note that tidyr also has a function called `extract` so make sure you mount the `raster` library after, or unmount `tidyr`, otherwise it may cause errors).
- What is the landslide susceptibility at this site? What would you tell the developer? Could you say anything about the uncertainty in landslide risk?
- Create a single visualization to communicate the risk of landslide at the site of interest (could be a map, plot or combination).