Some methods for the quantification of prediction uncertainties for digital soil mapping: Bootstrapping

Soil Security Laboratory

2018

1 Bootstrapping

Bootstrapping is a popular non-parametric approach for quantifying prediction uncertainties (Efron and Tibshirani, 1993). Bootstrapping involves repeated random sampling with replacement of the available data. With the bootstrap sample, a model is fitted, and can then be applied to generate a digital soil map. By repeating the process of random sampling and applying the model, we are able to generate probability distributions of the prediction realizations from each model at each pixel. A robust estimate may be determined by taking the average of all the simulated predictions at each pixel. By being able to obtain probability distributions of the outcomes, one is also able to quantify the uncertainty of the modeling by computing a prediction interval given a specified level of confidence. While the bootstrapping approach is relatively straightforward, there is a requirement to generate x number of maps, where x is the number of bootstrap samples. This obviously could be prohibitive from a computational and data storage point of view, but not altogether impossible (given parallel processing capabilities etc.) as was demonstrated by both Viscarra Rossel et al. (2015) and Liddicoat et al. (2015) whom both performed bootstrapping for quantification of uncertainties across very large mapping extents. In the case of Viscarra Rossel et al. (2015) this for for the entire Australian continent at 100m resolution.

In the example below, the bootstrap method is demonstrated. We will be using Cubist modeling for the model structure and perform 50 bootstrap samples. We will use 70% of the available data to use for fitting models. The remaining 30% as has been done for all previous DSM approaches is for validation. Of the 70% of the available data for model fitting, 70% of these data are to be used for each bootstrap sample i.e. a random sample with replacement.

1.1 Defining the model parameters

For the first step, we do the random partitioning of the data into calibration and validation data sets. Again we are using the `HV_subsoilpH` data and the

associated `hunterCovariates_sub` raster data stack.

```
## DATA Point data
data(HV_subsoilpH)
str(HV_subsoilpH)
```

```
## 'data.frame': 506 obs. of  14 variables:
##  $ X                       : num  340386 340345 340559 340483 340734 ...
##  $ Y                       : num  6368690 6368491 6369168 6368740 6368964 ...
##  $ pH60_100cm              : num  4.47 5.42 6.26 8.03 8.86 ...
##  $ Terrain_Ruggedness_Index: num  1.34 1.42 1.64 1.04 1.27 ...
##  $ AACN                    : num  1.619 0.281 2.301 1.74 3.114 ...
##  $ Landsat_Band1           : int  57 47 59 52 62 53 47 52 53 63 ...
##  $ Elevation               : num  103.1 103.7 99.9 101.9 99.8 ...
##  $ Hillshading             : num  1.849 1.428 0.934 1.517 1.652 ...
##  $ Light_insolation        : num  1689 1701 1722 1688 1735 ...
##  $ Mid_Slope_Positon       : num  0.876 0.914 0.844 0.848 0.833 ...
##  $ MRVBF                   : num  3.85 3.31 3.66 3.92 3.89 ...
##  $ NDVI                    : num  -0.143 -0.386 -0.197 -0.14 -0.15 ...
##  $ TWI                     : num  17.5 18.2 18.8 18 17.8 ...
##  $ Slope                   : num  1.79 1.42 1.01 1.49 1.83 ...
```

```
# Raster data
data(hunterCovariates_sub)
hunterCovariates_sub
```

```
## class       : RasterStack
## dimensions  : 249, 210, 52290, 11  (nrow, ncol, ncell, nlayers)
## resolution  : 25, 25  (x, y)
## extent      : 338422.3, 343672.3, 6364203, 6370428  (xmin, xmax, ymin, ymax)
## coord. ref. : +proj=utm +zone=56 +south +ellps=WGS84 +datum=WGS84 +units=m +no_defs
```

```
# subset data for modeling
set.seed(667)
training <- sample(nrow(HV_subsoilpH), 0.7 * nrow(HV_subsoilpH))
cDat <- HV_subsoilpH[training, ]
vDat <- HV_subsoilpH[-training, ]
```

The `nbag` variable below holds the value for the number of bootstrap models we want to fit. Here it is 50. Essentially the bootstrap can can be contained within a `for` loop, where upon each loop a sample of the available data is taken (here 70%) then a model is fitted. Note below the use of the `replace` parameter to indicate we want random sample with replacement. After a model is fitted, we save the model to file and will come back to it later. The `modelFile` variable shows the extensive use of the `paste` function in order to provide the pathway and file name for the model that we want to save on each loop iteration. The `saveRDS` function allows us to save each of the model objects as rds files to the location specified. An alternative to save the models individually to file is to save them to elements within a `list`. When dealing with very large numbers of models and additionally are complex in terms of their parameterizations, the save to `list` elements alternative could run into computer memory limitation issues. The last section of the script below just

demonstrates the use of the `list.files` functions to confirm that we have saved those models to file and they are ready to use.

```
# Number of bootstraps
nbag <- 50

# Fit cubist models for each bootstrap
library(Cubist)
for (i in 1:nbag) {
    trainingREP <- sample.int(nrow(cDat), 0.7 * nrow(cDat), replace = TRUE)

    fit_cubist <- cubist(x = cDat[trainingREP, c("Terrain_Ruggedness_Index",
        "AACN", "Landsat_Band1", "Elevation", "Hillshading", "Light_insolation",
        "Mid_Slope_Positon", "MRVBF", "NDVI", "TWI", "Slope")],
        y = cDat$pH60_100cm[trainingREP],
        cubistControl(rules = 5, extrapolation = 5), committees = 1)

    ### Note you will likely have different file path names ###
    modelFile <- paste(paste(paste(paste(getwd(), "/~/",
        sep = ""), "bootMod_", sep = ""), i, sep = ""), ".rds", sep = "")

    saveRDS(object = fit_cubist, file = modelFile)
}

# list all files in directory Note you will likely have different file path
# names ###
c.models <- list.files(path = paste(getwd(), "/~",
    sep = ""), pattern = "\\.rds$", full.names = TRUE)
```

We can then assess the goodness of fit and validation statistics of the bootstrap models. This is done using the `goof` function as in previous examples. This time we incorporate that function within a `for` loop. For each loop, we read in the model via the `radRDS` function and then save the diagnostics to the `cubiMat` matrix object. After the iterations are completed, we use the `colMeans` function to calculate the means of the diagnostics over the 50 model iterations. You could also assess the variance of those means by a command such as `var(cubiDat[,1])`, which would return the variance of the R^2 values.

```
# calibration data
cubiMat <- matrix(NA, nrow = nbag, ncol = 5)
for (i in 1:nbag) {
    fit_cubist <- readRDS(c.models[i])
    cubiMat[i, ] <- as.matrix(goof(observed = cDat$pH60_100cm,
    predicted = predict(fit_cubist,
        newdata = cDat)))
}
cubiDat <- as.data.frame(cubiMat)
names(cubiDat) <- c("R2", "concordance", "MSE", "RMSE", "bias")
colMeans(cubiDat)
```

```
##          R2 concordance          MSE        RMSE        bias
##  0.25261147  0.45185565  1.46214146  1.20887737 -0.06697598
```

```
# Validation data
cubPred.V <- matrix(NA, ncol = nbag, nrow = nrow(vDat))
cubiMat <- matrix(NA, nrow = nbag, ncol = 5)
for (i in 1:nbag) {
    fit_cubist <- readRDS(c.models[i])
    cubPred.V[, i] <- predict(fit_cubist, newdata = vDat)
    cubiMat[i, ] <- as.matrix(goof(observed = vDat$pH60_100cm,
     predicted = predict(fit_cubist,
        newdata = vDat)))
}
cubPred.V_mean <- rowMeans(cubPred.V)

cubiDat <- as.data.frame(cubiMat)
names(cubiDat) <- c("R2", "concordance", "MSE", "RMSE", "bias")
colMeans(cubiDat)
```

```
##          R2 concordance          MSE        RMSE        bias
##  0.09010054  0.26690013  1.80777927  1.34013203  0.11262625
```

```
# Average validation MSE
avGMSE <- mean(cubiDat[, 3])
```

For the validation data, in addition to deriving the model diagnostic statistics, we are also saving the actual model predictions for these data for each iteration to the `cubPred.V` object. These will be used further on for validating the prediction uncertainties. The last line of the script above saves the mean of the mean square error (MSE) estimates from the validation data. The independent MSE estimator, accounts for both systematic and random errors in the modeling. This estimate of the MSE is needed for quantifying the uncertainties, as this error is in addition to that which are accounted for by the bootstrap, which are specifically those associated with the deterministic model component i.e. the model relationship between target variable and the covariates. Subsequently an overall prediction variance (at each point or pixel) will be the sum of the random error component (MSE) and the bootstrap prediction variance (as estimated from the mean of the realisations from the bootstrap modeling).

1.2 Mapping

Our initial purpose here is to derive the mean and the variance of the predictions from each bootstrap sample. This requires loading in each bootstrap model, applying into the covariate data, then saving the predicted map to file or R memory. In the case below the predictions are saved to file. This is illustrated in the following script.

```
### Note you will likely have different file path names ###
for (i in 1:nbag) {
    fit_cubist <- readRDS(c.models[i])
```

```
    mapFile <- paste(paste(paste(paste(getwd(), "/~/",
        sep = ""), "bootMap_", sep = ""), i, sep = ""), ".tif", sep = "")
    predict(hunterCovariates_sub, fit_cubist, filename = mapFile, format = "GTiff",
        overwrite = T)
}
```

To evaluate the mean at each pixel from each of the created maps, the base function **mean** can be applied to a given stack of rasters. First we need to get the path location of the rasters. Notice from the **list.files** function and the **pattern** parameter, we are restricting the search of rasters that contain the string "bootMap". Next we make a stack of those rasters, followed by the calculation of the mean, which is also written directly to file.

```
# Pathway to rasters Note you will likely have different file path names ###
files <- list.files(paste(getwd(), "/~/",
    sep = ""), pattern = "bootMap", full.names = TRUE)

# Raster stack
r1 <- raster(files[1])
for (i in 2:length(files)) {
    r1 <- stack(r1, files[i])
}

# Calculate mean
meanFile <- paste(paste(paste(getwd(), "/~/",
    sep = ""), "meanPred_", sep = ""), ".tif", sep = "")
bootMap.mean <- writeRaster(mean(r1), filename = meanFile, format = "GTiff",
    overwrite = TRUE)
```

There is not a simple **R** function to use in order to estimate the variance at each pixel from the prediction maps. Therefore we resort to estimating it directly from the standard equation:

$$Var(X) = \frac{1}{1-n}\sum_{i=1}^{n}(x_i - \mu)^2 \tag{1}$$

The symbol μ in this case is the mean bootstrap prediction, and x_i is the *ith* bootstrap map. In the first step below, we estimate the square differences and save the maps to file. Then we calculate the sum of those squared differences, before deriving the variance prediction. The last step is to add the variance of the bootstrap predictions to the averaged MSE estimated from the validation data.

```
# Square differences
for (i in 1:length(files)) {
    r1 <- raster(files[i])
    diffFile <- paste(paste(paste(paste(getwd(), "/~/",
        sep = ""), "bootAbsDif_", sep = ""), i, sep = ""), ".tif", sep = "")
    jj <- (r1 - bootMap.mean)^2
    writeRaster(jj, filename = diffFile, format = "GTiff", overwrite = TRUE)
```

```
}

# calculate the sum of square differences Look for files with the bootAbsDif
# character string in file name
files2 <- list.files(paste(getwd(), "/~/",
    sep = ""), pattern = "bootAbsDif", full.names = TRUE)
# stack
r2 <- raster(files2[1])
for (i in 2:length(files2)) {
    r2 <- stack(r1, files2[i])
}

sqDiffFile <- paste(paste(paste(getwd(), "/~/",
    sep = ""), "sqDiffPred_", sep = ""), ".tif", sep = "")
bootMap.sqDiff <- writeRaster(sum(r2), filename = sqDiffFile, format = "GTiff",
    overwrite = TRUE)


# Variance
varFile <- paste(paste(paste(getwd(), "/~/",
    sep = ""), "varPred_", sep = ""), ".tif", sep = "")
bootMap.var <- writeRaster(((1/(nbag - 1)) * bootMap.sqDiff), filename = varFile,
    format = "GTiff", overwrite = TRUE)

# Overall prediction variance
varFile2 <- paste(paste(paste(getwd(), "/~/",
    sep = ""), "varPredF_", sep = ""), ".tif", sep = "")
bootMap.varF <- writeRaster((bootMap.var + avGMSE), filename = varFile, format = "GTiff",
    overwrite = TRUE)
```

To derive to 90% prediction interval we take the square root of the variance estimate and multiply that value by the z value that corresponds to a 90% probability. The z value is obtained using the `qnorm` function. The result is then either added or subtracted to the mean prediction in order to generate the upper and lower prediction limits respectively.

```
# Standard deviation
sdFile <- paste(paste(paste(getwd(), "/~/",
    sep = ""), "sdPred_", sep = ""), ".tif", sep = "")
bootMap.sd <- writeRaster(sqrt(bootMap.varF), filename = sdFile, format = "GTiff",
    overwrite = TRUE)

# standard error
seFile <- paste(paste(paste(getwd(), "/~/",
    sep = ""), "sePred_", sep = ""), ".tif", sep = "")
bootMap.se <- writeRaster((bootMap.sd * qnorm(0.95)), filename = seFile, format = "GTiff",
    overwrite = TRUE)


# upper prediction limit
```

```
uplFile <- paste(paste(paste(getwd(), "~/",
    sep = ""), "uplPred_", sep = ""), ".tif", sep = "")
bootMap.upl <- writeRaster((bootMap.mean + bootMap.se), filename = uplFile,
    format = "GTiff", overwrite = TRUE)

# lower prediction limit
lplFile <- paste(paste(paste(getwd(), "/~/",
    sep = ""), "lplPred_", sep = ""), ".tif", sep = "")
bootMap.lpl <- writeRaster((bootMap.mean - bootMap.se), filename = lplFile,
    format = "GTiff", overwrite = TRUE)

# prediction interval range
pirFile <- paste(paste(paste(getwd(), "/~/",
    sep = ""), "pirPred_", sep = ""), ".tif", sep = "")
bootMap.pir <- writeRaster((bootMap.upl - bootMap.lpl), filename = pirFile,
    format = "GTiff", overwrite = TRUE)
```

As for the Universal kriging example, we can plot the associated maps of the predictions and quantified uncertainties (Figure 1.

```
phCramp <- c("#d53e4f", "#f46d43", "#fdae61", "#fee08b", "#ffffbf", "#e6f598",
    "#abdda4", "#66c2a5", "#3288bd", "#5e4fa2", "#542788", "#2d004b")
brk <- c(2:14)
par(mfrow = c(2, 2))
plot(bootMap.lpl, main = "90% Lower prediction limit", breaks = brk, col = phCramp)
plot(bootMap.mean, main = "Prediction", breaks = brk, col = phCramp)
plot(bootMap.upl, main = "90% Upper prediction limit", breaks = brk, col = phCramp)
plot(bootMap.pir, main = "Prediction limit range", col = terrain.colors(length(seq(0,
    6.5, by = 1)) - 1), axes = FALSE, breaks = seq(0, 6.5, by = 1))
```

1.3 Validating the quantification of uncertainty

You will recall the bootstrap model predictions on the validation data were saved to the `cubPred.V` object. We want estimate the standard deviation of those predictions for each point. Also recall that the prediction variance is the sum of the MSE and the bootstrap models prediction variance. Taking the square root of that summation results in standard deviation estimate.

```
val.sd <- matrix(NA, ncol = 1, nrow = nrow(cubPred.V))
for (i in 1:nrow(cubPred.V)) {
    val.sd[i, 1] <- sqrt(var(cubPred.V[i, ]) + avGMSE)
}
```

We then need to multiply the standard deviation by the corresponding percentile of the standard normal distribution in order to express the prediction limits at each level of confidence. Note the use of the `for` loop and the associated cycling through of the different percentile values.

```
# Percentiles of normal distribution
qp <- qnorm(c(0.995, 0.9875, 0.975, 0.95, 0.9, 0.8, 0.7, 0.6, 0.55, 0.525))
```

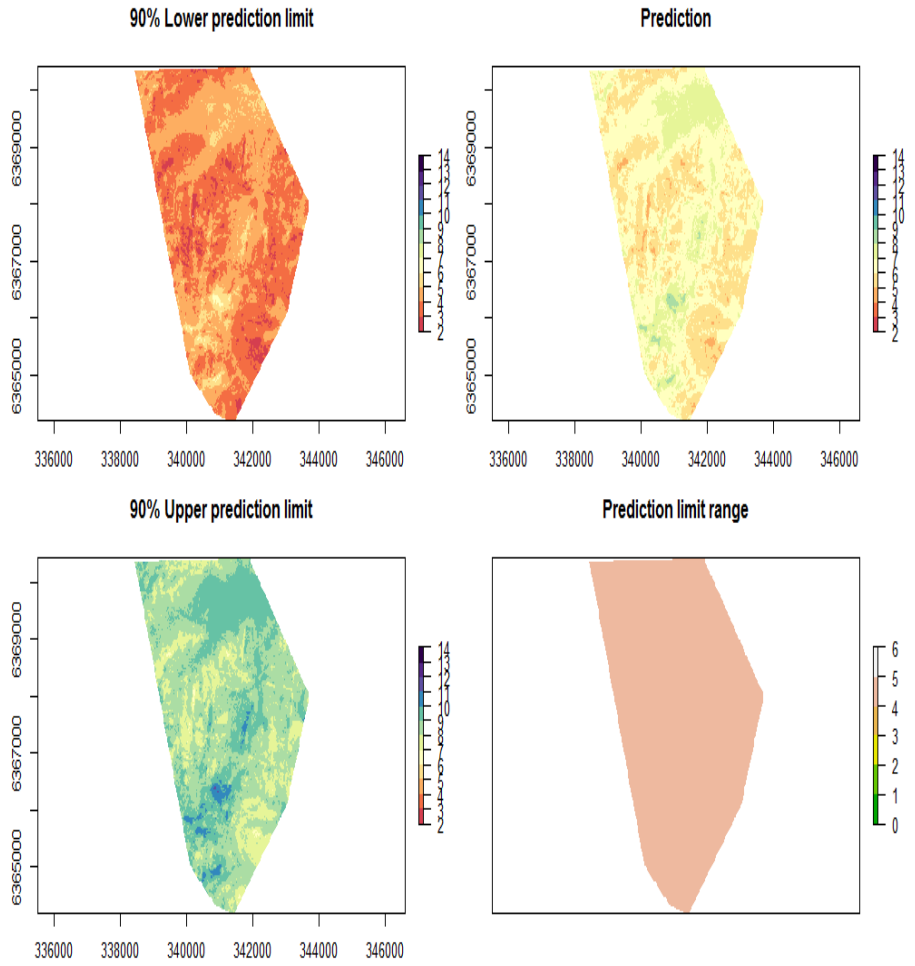

Figure 1: Soil pH predictions and prediction limits derived using bootstrapping.

```
# zfactor multiplication
vMat <- matrix(NA, nrow = nrow(cubPred.V), ncol = length(qp))
for (i in 1:length(qp)) {
    vMat[, i] <- val.sd * qp[i]
}
```

Now we add or subtract the limits to/from the averaged model predictions to derive to prediction limits for each level of confidence.

```
# upper prediction limit
uMat <- matrix(NA, nrow = nrow(cubPred.V), ncol = length(qp))
for (i in 1:length(qp)) {
    uMat[, i] <- cubPred.V_mean + vMat[, i]
}
```

```
# lower prediction limit
lMat <- matrix(NA, nrow = nrow(cubPred.V), ncol = length(qp))
for (i in 1:length(qp)) {
    lMat[, i] <- cubPred.V_mean - vMat[, i]
}
```

Now we assess the PICP for each level confidence. Recalling that we are simply assessing whether the observed value is encapsulated by the corresponding prediction limits, then calculating the proportion of agreement to total number of observations.

```
bMat <- matrix(NA, nrow = nrow(cubPred.V), ncol = length(qp))
for (i in 1:ncol(bMat)) {
    bMat[, i] <- as.numeric(vDat$pH60_100cm <= uMat[, i] & vDat$pH60_100cm >=
        lMat[, i])
}
```

```
colSums(bMat)/nrow(bMat)
```

```
##  [1] 1.00000000 1.00000000 0.99342105 0.96710526 0.90131579 0.69078947
##  [7] 0.45394737 0.21052632 0.08552632 0.03289474
```

As can be seen on Figure 2, there is an indication that the prediction uncertainties could be a little too liberally defined, where particularly at the higher level of confidence the associated PICP is higher.

```
# make plot
cs <- c(99, 97.5, 95, 90, 80, 60, 40, 20, 10, 5)  # confidence level
plot(cs, ((colSums(bMat)/nrow(bMat)) * 100))
```

Quantiles of the distribution of the prediction limit range are express below for the validation data (in terms of the 90% level of confidence). Compared to the universal kriging approach, the uncertainties quantified from the bootstrapping approach are higher in general.

```
cs <- c(99, 97.5, 95, 90, 80, 60, 40, 20, 10, 5)  # confidence level
colnames(lMat) <- cs
colnames(uMat) <- cs
quantile(uMat[, "90"] - lMat[, "90"])
```

```
##       0%      25%      50%      75%     100%
## 4.551972 4.702860 4.795511 4.950161 6.610331
```

References

Efron, B. and R. Tibshirani
1993. *An Introduction to the Bootstrap*. London: Chapman and Hall.

Liddicoat, C., D. Maschmedt, D. Clifford, R. Searle, T. Herrmann, L. Macdonald, and J. Baldock
2015. Predictive mapping of soil organic carbon stocks in south australias agricultural zone. *Soil Research*, 53:956–973.

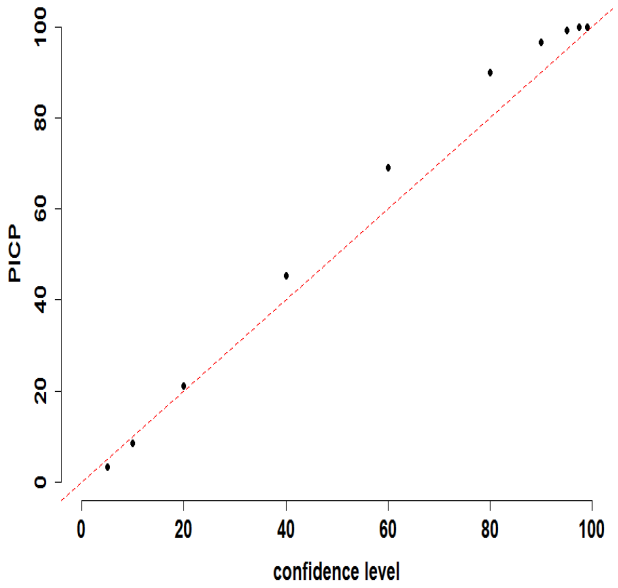

Figure 2: Plot of PICP and confidence level based on validation of bootstrapping model.

Viscarra Rossel, R. A., C. Chen, M. J. Grundy, R. Searle, D. Clifford, and P. H. Campbell
2015. The australian three-dimensional soil grid: Australias contribution to the globalsoilmap project. *Soil Research*, 53:845–864.