Some methods for the quantification of prediction uncertainties for digital soil mapping: Data partitioning and cross validation approach.

Soil Security Laboratory

2018

1 Empirical uncertainty quantification through data partitioning and cross validation.

The next two approaches for uncertainty quantification are empirical methods whereby the prediction intervals are defined from the distribution of model errors i.e. the deviation between observation and model predictions. In the examples to follow however, the prediction limits are not spatially uniform. Rather they are a function of the landscape. For example, in some areas of the landscape, or in some particular landscape situations, the prediction is going to be more accurate than in other situations. For the immediate approach, an explicit landscape partition is used to define likely areas where the quantification of uncertainty may differ. This approach uses the Cubist data mining algorithm for prediction and partitioning of the landscape. Within each partitioned area, we can then define the distribution of model errors. This approach for uncertainty quantification was used by Malone et al. (2014) where soil pH was predicted using a Cubist model and residual kriging spatial model.

This approach and the next are useful from the viewpoint that complex models (in terms of parameterisation) are able to be used within a digital soil mapping framework where there is a necessity to define a stated level of prediction confidence. With universal kriging, this is the restriction of the linear model (which in many situations could be the best model). For empirical approaches to uncertainty quantification, this restriction does not exist. Further to this, the localized allocation of prediction limits i.e. in given landscape situations provides a means to identify areas where mapping is accurate and where it is not so much. Often this is a function of the sampling density. This explicit allocation of prediction intervals may assist with the future design of soil sampling programs.

In the approach to be described the partitioning is a “hard” partition, for the next approach described in the next section, it is a “soft” or “fuzzy” partition.

1.1 Defining the model parameters

We will use a Cubist model regression kriging approach as used previously, and then define the subsequent prediction intervals.

First we prepare the data.

```
# Library
library(ithir)
library(sp)
library(rgdal)
library(raster)
library(gstat)

## 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, ]
```

Now we fit the Cubist model using all available environmental covariates.

```r
library(Cubist)

# Cubist model
hv.cub.Exp <- cubist(x = cDat[, c("Terrain_Ruggedness_Index", "AACN", "Landsat_Band1",
    "Elevation", "Hillshading", "Light_insolation", "Mid_Slope_Positon", "MRVBF",
    "NDVI", "TWI", "Slope")], y = cDat$pH60_100cm, cubistControl(unbiased = TRUE,
    rules = 100, extrapolation = 10, sample = 0, label = "outcome"), committees = 1)
summary(hv.cub.Exp)
```

```
##
## Call:
## cubist.default(x = cDat[, c("Terrain_Ruggedness_Index",
##  = cubistControl(unbiased = TRUE, rules = 100, extrapolation = 10,
##  sample = 0, label = "outcome"))
##
##
## Cubist [Release 2.07 GPL Edition]  Thu Dec 28 22:47:30 2017
## -------------------------------
##
##     Target attribute `outcome'
##
## Read 354 cases (12 attributes) from undefined.data
##
## Model:
##
##   Rule 1: [116 cases, mean 6.183380, range 3.956412 to 9.249626, est err 0.714710]
##
##     if
##  NDVI <= -0.191489
##  TWI <= 13.17387
##     then
##  outcome = 9.610208 + 0.0548 AACN - 0.0335 Elevation + 0.131 Hillshading
##            + 3.5 NDVI + 0.076 Terrain_Ruggedness_Index - 0.055 Slope
##            - 0.023 Landsat_Band1 + 0.03 MRVBF + 0.07 Mid_Slope_Positon
##            + 0.005 TWI
##
##   Rule 2: [164 cases, mean 6.533212, range 3.437355 to 9.741758, est err 0.986175]
##
##     if
##  TWI > 13.17387
##     then
##  outcome = 7.471082 + 0.0215 AACN + 0.108 Hillshading + 4.2 NDVI
##            + 0.24 MRVBF + 1.16 Mid_Slope_Positon - 0.0104 Elevation
##            - 0.069 Slope + 0.077 Terrain_Ruggedness_Index
##            - 0.028 Landsat_Band1 + 0.047 TWI
##
##   Rule 3: [74 cases, mean 6.926269, range 2.997182 to 9.630296, est err 1.115631]
##
##     if
```

```
##  NDVI > -0.191489
##  TWI <= 13.17387
##     then
##  outcome = 11.743466 + 0.0416 AACN - 0.091 Landsat_Band1
##            - 0.0117 Elevation + 3 NDVI + 0.048 Hillshading
##            + 0.065 Terrain_Ruggedness_Index - 0.046 Slope
##
##
## Evaluation on training data (354 cases):
##
##     Average  |error|           1.085791
##     Relative |error|               0.95
##     Correlation coefficient        0.44
##
##
##  Attribute usage:
##    Conds  Model
##
##    100%    79%    TWI
##     54%   100%    NDVI
##           100%    Terrain_Ruggedness_Index
##           100%    AACN
##           100%    Landsat_Band1
##           100%    Elevation
##           100%    Hillshading
##           100%    Slope
##            79%    Mid_Slope_Positon
##            79%    MRVBF
##
##
## Time: 0.0 secs
```

The `hv.cub.Exp` model indicates three subsets in the data were made. Assessing the goodness of fit of this model, we use the `goof` function. This model seems to perform OK.

```
goof(observed = cDat$pH60_100cm, predicted = predict(hv.cub.Exp, newdata = cDat))
```

```
##          R2 concordance      MSE     RMSE         bias
## 1 0.3247524   0.4779242 1.266316 1.125307 2.640776e-07
```

Now we want to assess the model residuals for spatial auto correlation. In this case we need to look at the variogram of the residuals.

```
cDat$residual <- cDat$pH60_100cm - predict(hv.cub.Exp, newdata = cDat)

# coordinates
coordinates(cDat) <- ~X + Y
# residual variogram model
vgm1 <- variogram(residual ~ 1, cDat, width = 200)
mod <- vgm(psill = var(cDat$residual), "Sph", range = 10000, nugget = 0)
```

```
model_1 <- fit.variogram(vgm1, mod)

gOK <- gstat(NULL, "hunterpH_cubistRES", residual ~ 1, cDat, model = model_1)
gOK
```

```
## data:
## hunterpH_cubistRES : formula = residual`~`1 ; data dim = 354 x 13
## variograms:
##                         model     psill    range
## hunterpH_cubistRES[1]   Nug 0.7430966   0.0000
## hunterpH_cubistRES[2]   Sph 0.5385924 937.3458
```

This output indicates there is a reasonable variogram of the residuals with which may help improve the overall predictive model. We can determine whether there is any improvement later when we perform the validation.

With a model defined, it is now necessary to estimate the within partition prediction limits. This is done using a leave-one-out cross validation procedure using a Cubist model regression kriging model as we did for the spatial model. What we need to do first is to determine which observations in the data set belong to which partition. Looking at the `summary` output above we can examine the threshold criteria that define the data partitions.

```
# Assign a rule to each observation if more than one observation
cDat1 <- as.data.frame(cDat)
cDat1$rule = 9999
# rule 1
cDat1$rule[which(cDat1$NDVI <= -0.191489 & cDat1$TWI <= 13.17387)] <- 1
# rule 2
cDat1$rule[which(cDat1$TWI > 13.17387)] <- 2
## rule 3
cDat1$rule[which(cDat1$NDVI > -0.191489 & cDat1$TWI <= 13.17387)] <- 3

cDat1$rule <- as.factor(cDat1$rule)
summary(cDat1$rule)
```

```
##   1   2   3
## 115 165  74
```

Now we can subset `cDat1` based on its defined rule or partition and then perform the LOCV. The script below shows the procedure for the process using the first rule.

```
# subset the data
cDat1.r1 <- cDat1[which(cDat1$rule == 1), ]
target.C.r1 <- cDat1.r1$pH60_100cm

########## Leave-one-out cross validation ###################
looResiduals <- numeric(nrow(cDat1.r1))
for (i in 1:nrow(cDat1.r1)) {
    loocubistPred <- cubist(x = cDat1.r1[-i, 4:14], y = target.C.r1[-i],
    cubistControl(unbiased = F,
```

```
        rules = 1, extrapolation = 5, sample = 0, seed = sample.int(4096, size = 1) -
            1L, label = "outcome"), committees = 1)  # fit cubist model
    # fit cubist model
    cDat11.r1.sub <- cDat1.r1[-i, ]
    cDat11.r1.sub$pred <- predict(loocubistPred, newdata = cDat11.r1.sub, neighbors = 0)
    cDat11.r1.sub$resids <- target.C.r1[-i] - cDat11.r1.sub$pred
    # residual variogram
    vgm.r1 <- variogram(resids ~ 1, ~X + Y, cDat11.r1.sub, width = 100)
    mod.r1 <- vgm(psill = var(cDat11.r1.sub$resids), "Sph", range = 10000, nugget = 0)
    model_1.r1 <- fit.variogram(vgm.r1, mod.r1)
    # interpolate residual on withheld point
    int.resids1.r1 <- krige(cDat11.r1.sub$resids ~ 1, locations = ~X + Y,
    data = cDat11.r1.sub,
        newdata = cDat1.r1[i, c("X", "Y")], model = model_1.r1)[, 3]
    # Cubist model predict on withheld point
    looPred <- predict(loocubistPred, newdata = cDat1.r1[i, ], neighbors = 0)
    # Add
    looResiduals[i] <- target.C.r1[i] - (looPred + int.resids1.r1)
}
```

We are interested in evaluating the 90% prediction interval. We do this for each data partition by taking the lower 5^{th} and upper 95^{th} quantiles of the residual distribution, and ultimately add these to the model predictions. For the validation, we also have to evaluate the quantiles for a range of confidence levels (in this case sequentially from 5% to 99%) for calculation of the PICP. This is done using the `quantile` function.

```
# Rule 1 90% confidence
r1.ulPI <- quantile(looResiduals1, probs = c(0.05, 0.95), na.rm = FALSE, names = F,
    type = 7)  # rule lower and upper PI
r1.ulPI
```

```
## [1] -1.210314  1.762870
```

```
# Confidence interval range
r1.q <- quantile(looResiduals1, probs = c(0.005, 0.995, 0.0125, 0.9875, 0.025,
    0.975, 0.05, 0.95, 0.1, 0.9, 0.2, 0.8, 0.3, 0.7, 0.4, 0.6, 0.45, 0.55, 0.475,
    0.525), na.rm = FALSE, names = F, type = 7)

# Rule 2 90% confidence
r2.ulPI <- quantile(looResiduals2, probs = c(0.05, 0.95), na.rm = FALSE, names = F,
    type = 7)  # rule lower and upper PI
r2.ulPI
```

```
## [1] -2.429566  2.414661
```

```
# Confidence interval range
r2.q <- quantile(looResiduals2, probs = c(0.005, 0.995, 0.0125, 0.9875, 0.025,
    0.975, 0.05, 0.95, 0.1, 0.9, 0.2, 0.8, 0.3, 0.7, 0.4, 0.6, 0.45, 0.55, 0.475,
    0.525), na.rm = FALSE, names = F, type = 7)

# Rule 3 90% confidence
```

```
r3.ulPI <- quantile(looResiduals3, probs = c(0.05, 0.95), na.rm = FALSE, names = F,
    type = 7)  # rule lower and upper PI
r3.ulPI
```

```
## [1] -1.845151  1.678207
```

```
# Confidence interval range
r3.q <- quantile(looResiduals3, probs = c(0.005, 0.995, 0.0125, 0.9875, 0.025,
    0.975, 0.05, 0.95, 0.1, 0.9, 0.2, 0.8, 0.3, 0.7, 0.4, 0.6, 0.45, 0.55, 0.475,
    0.525), na.rm = FALSE, names = F, type = 7)
```

As can be seen, the prediction limits for each partition of the data are quite different from each other.

1.2 Spatial mapping

To create the maps in the same fashion as was done for bootstrapping and universal kriging, we do more-or-less as the same for the Cubist regression kriging. First create the regression kriging map.

```
# map
map.cubist <- predict(hunterCovariates_sub, hv.cub.Exp, args = list(neighbors = 0),
    filename = "rk_cubist.tif", format = "GTiff", overwrite = T)

# kriged residuals
map.cubist.res <- interpolate(hunterCovariates_sub, gOK, xyOnly = TRUE, index = 1,
    filename = "rk_residuals.tif", format = "GTiff", datatype = "FLT4S",
    overwrite = TRUE)

# raster stack of predictions and residuals
r2 <- stack(map.cubist, map.cubist.res)
f1 <- function(x) calc(x, sum)
map.cubist.final <- calc(r2, fun = sum, filename = "cubistRK.tif", format = "GTiff",
    overwrite = T)
```

To derive the upper and lower prediction limits we can apply the raster calculations in a very manual way, such that each line of the `rasterStack` is read in and then evaluated to determine which rule each entry on the line belongs to. Then given the rule, the corresponding upper and lower limits are appended to a new raster together with a raster which indicates which rule was applied where. For small raster data sets where the mapping extent is small, this approach works fine. It can also be applied for very large rasters, but can take some time as the reading of the raster occurs line by line. There are however, raster options to process them this way in chunk form.

```
# Create new raster datasets
upper1 <- raster(hunterCovariates_sub[[1]])
lower1 <- raster(hunterCovariates_sub[[1]])
rule1 <- raster(hunterCovariates_sub[[1]])
upper1 <- writeStart(upper1, filename = "cubRK_upper1.tif", format = "GTiff",
    overwrite = TRUE)
```

```
lower1 <- writeStart(lower1, filename = "cubRK_lower1.tif", format = "GTiff",
    overwrite = TRUE)
rule1 <- writeStart(rule1, filename = "cubRK_rule1.tif", format = "GTiff",
datatype = "INT2S", overwrite = TRUE)

for (i in 1:dim(upper1)[1]) {
    # extract raster information line by line
    cov.Frame <- as.data.frame(getValues(hunterCovariates_sub, i))
    ulr.Frame <- matrix(NA, ncol = 3, nrow = dim(upper1)[2])
    # append in partition information

    # rule 1
    ulr.Frame[which(cov.Frame$NDVI <= -0.191489 & cov.Frame$TWI <= 13.17387),
        1] <- r1.ulPI[2]
    ulr.Frame[which(cov.Frame$NDVI <= -0.191489 & cov.Frame$TWI <= 13.17387),
        2] <- r1.ulPI[1]
    ulr.Frame[which(cov.Frame$NDVI <= -0.191489 & cov.Frame$TWI <= 13.17387),
        3] <- 1
    # rule 2
    ulr.Frame[which(cov.Frame$TWI > 13.17387), 1] <- r2.ulPI[2]
    ulr.Frame[which(cov.Frame$TWI > 13.17387), 2] <- r2.ulPI[1]
    ulr.Frame[which(cov.Frame$TWI > 13.17387), 3] <- 2
    # rule 3
    ulr.Frame[which(cov.Frame$NDVI > -0.191489 & cov.Frame$TWI <= 13.17387),
        1] <- r3.ulPI[2]
    ulr.Frame[which(cov.Frame$NDVI > -0.191489 & cov.Frame$TWI <= 13.17387),
        2] <- r3.ulPI[1]
    ulr.Frame[which(cov.Frame$NDVI > -0.191489 & cov.Frame$TWI <= 13.17387),
        3] <- 3

    ulr.Frame <- as.data.frame(ulr.Frame)
    names(ulr.Frame) <- c("upper", "lower", "rule")
    # write to raster then close
    pred_upper <- ulr.Frame$upper
    pred_lower <- ulr.Frame$lower
    pred_rule <- ulr.Frame$rule
    upper1 <- writeValues(upper1, pred_upper, i)
    lower1 <- writeValues(lower1, pred_lower, i)
    rule1 <- writeValues(rule1, pred_rule, i)
    print(i)
}
upper1 <- writeStop(upper1)
lower1 <- writeStop(lower1)
rule1 <- writeStop(rule1)
```

Now we can derive the prediction interval by adding the upper and lower limits to the regression kriging prediction that was made earlier. Then we can estimate the prediction interval range.

```
# raster stack of predictions and prediction limits
r2 <- stack(map.cubist.final, lower1)  #lower
mapRK.lower <- calc(r2, fun = sum, filename = "cubistRK_lowerPL.tif", format = "GTiff",
    overwrite = T)


# raster stack of predictions and prediction limits
r2 <- stack(map.cubist.final, upper1)  #upper
mapRK.upper <- calc(r2, fun = sum, filename = "cubistRK_upperPI.tif", format = "GTiff",
    overwrite = T)


# Prediction interval range
r2 <- stack(mapRK.lower, mapRK.upper)  #diff
mapRK.PIrange <- calc(r2, fun = diff, filename = "cubistRK_PIrange.tif",
format = "GTiff", overwrite = T)
```

Now we can plot the maps as before. We can note this time that the prediction interval range is smaller for the cubist regression kriging than it is for the universal kriging approach (Figure 1).

```
# color ramp
phCramp <- c("#d53e4f", "#f46d43", "#fdae61", "#fee08b", "#ffffbf", "#e6f598",
    "#abdda4", "#66c2a5", "#3288bd", "#5e4fa2", "#542788", "#2d004b")
brk <- c(2:14)
par(mfrow = c(2, 2))
plot(mapRK.lower, main = "90% Lower prediction limit", breaks = brk, col = phCramp)
plot(map.cubist.final, main = "Prediction", breaks = brk, col = phCramp)
plot(mapRK.upper, main = "90% Upper prediction limit", breaks = brk, col = phCramp)
plot(mapRK.PIrange, 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

For validation we assess both the quality of the predictions and the quantification of uncertainty as was done earlier for the universal kriging. Below we assess the validation of the cubist model alone and the cubist regression kriging model.

```
# Cubist model prediction
vPreds <- predict(hv.cub.Exp, newdata = vDat)
# Residual prediction
coordinates(vDat) <- ~X + Y
OK.preds.V <- as.data.frame(krige(residual ~ 1, cDat, model = model_1, newdata = vDat))
```

```
## [using ordinary kriging]
```

```
# Regression kriging predictions
OK.preds.V$cubist <- vPreds
OK.preds.V$finalP <- OK.preds.V$cubist + OK.preds.V$var1.pred
```

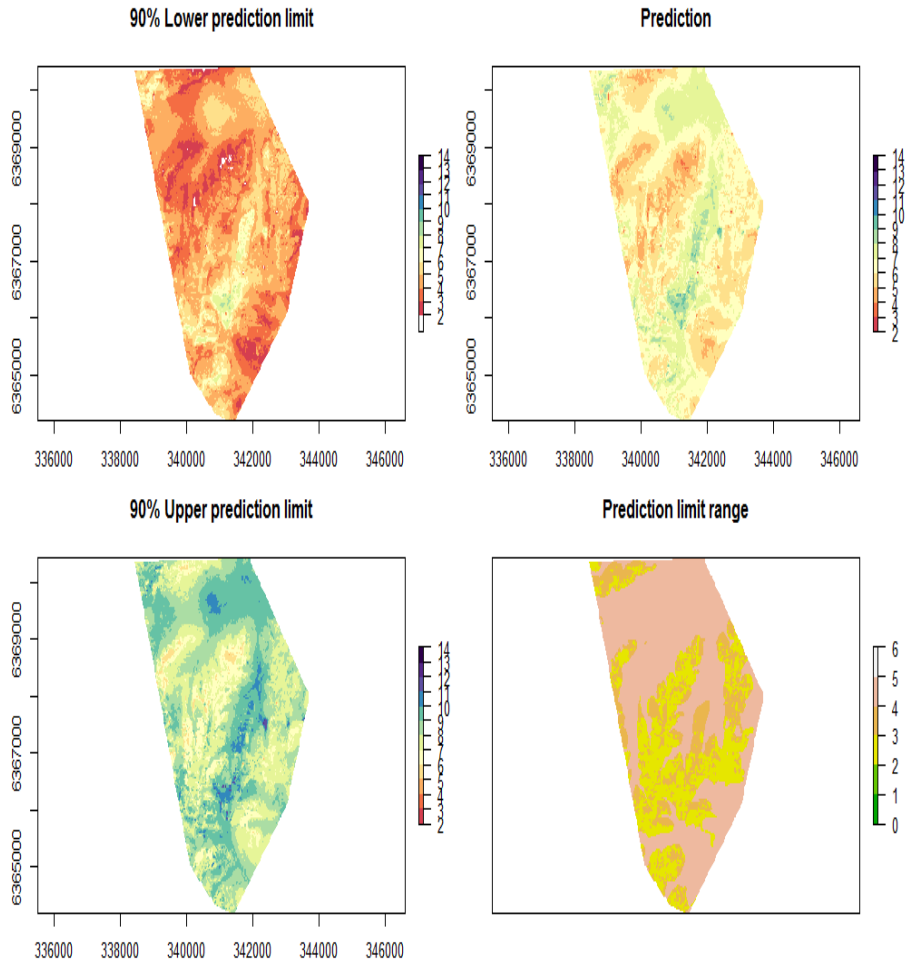

Figure 1: Soil pH predictions and prediction limits derived using a Cubist regression kriging model

```
# Validation regression
goof(observed = vDat$pH60_100cm, predicted = OK.preds.V$cubist)
```

```
##          R2 concordance      MSE     RMSE      bias
## 1 0.2218322    0.422689 1.352345 1.162903 0.2017115
```

```
# regression kriging
goof(observed = vDat$pH60_100cm, predicted = OK.preds.V$finalP)
```

```
##          R2 concordance     MSE    RMSE      bias
## 1 0.2951745    0.527646 1.25976 1.12239 0.1083254
```

The regression kriging model is a little more accurate than the Cubist model alone.

The first step in estimating the uncertainty about the validation points is to

evaluate with rule set or partition it belongs to.

```
# Assign a rule to each observation if more than one obervation
vDat1 <- as.data.frame(vDat)
# Insert for rules bit
vDat1$rule = 9999
# rule 1
vDat1$rule[which(vDat1$NDVI <= -0.191489 & vDat1$TWI <= 13.17387)] <- 1
# rule 2
vDat1$rule[which(vDat1$TWI > 13.17387)] <- 2
## rule 3
vDat1$rule[which(vDat1$NDVI > -0.191489 & vDat1$TWI <= 13.17387)] <- 3

vDat1$rule <- as.factor(vDat1$rule)
summary(vDat1$rule)
```

```
##  1  2  3
## 43 65 44
```

```
# append regression kriging predictions
vDat1 <- cbind(vDat1, OK.preds.V[, "finalP"])
names(vDat1)[ncol(vDat1)] <- "RKpred"
```

Then we can define the prediction interval that corresponds to each observation for each level of confidence.

```
# Upper PL
ulMat <- matrix(NA, nrow = nrow(vDat1), ncol = length(r1.q))
for (i in seq(2, 20, 2)) {
    ulMat[which(vDat1$rule == 1), i] <- r1.q[i]
    ulMat[which(vDat1$rule == 2), i] <- r2.q[i]
    ulMat[which(vDat1$rule == 3), i] <- r3.q[i]
}

# Lower PL
for (i in seq(1, 20, 2)) {
    ulMat[which(vDat1$rule == 1), i] <- r1.q[i]
    ulMat[which(vDat1$rule == 2), i] <- r2.q[i]
    ulMat[which(vDat1$rule == 3), i] <- r3.q[i]
}

# upper and lower prediction limits
ULpreds <- ulMat + vDat1$RKpred

# binary
bMat <- matrix(NA, nrow = nrow(ULpreds), ncol = (ncol(ULpreds)/2))
cnt <- 1
for (i in seq(1, 20, 2)) {
    bMat[, cnt] <- as.numeric(vDat1$pH60_100cm <= ULpreds[, i + 1] & vDat1$pH60_100cm >=
        ULpreds[, i])
    cnt <- cnt + 1
}
```

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

```
##  [1] 0.99342105 0.98026316 0.96052632 0.90789474 0.81578947 0.59868421
##  [7] 0.46052632 0.25657895 0.14473684 0.09210526
```

The PICP estimates appear to correspond quite well with the respective confidence levels. This can be observed from the plot on Figure 2 too.

```
# make plot
cs <- c(99, 97.5, 95, 90, 80, 60, 40, 20, 10, 5)  # confidence level
plot(cs, ((colSums(bMat)/nrow(bMat)) * 100))
abline(a = 0, b = 1, lty = 2, col = "red")
```

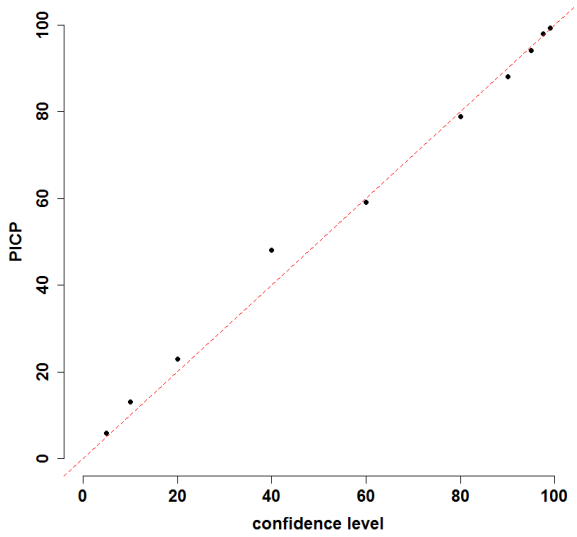

Figure 2: Plot of PICP and confidence level based on validation of Cubist regression kriging model.

From the validation observations the prediction intervals range between 2.45 and 4.6 with a median of about 3.36 pH units when using the Cubist regression kriging model.

```
quantile(ULpreds[, 8] - ULpreds[, 7])
```

```
##       0%      25%      50%      75%     100%
## 2.973184 2.973184 3.523359 4.844228 4.844228
```

References

Malone, B. P., B. Minasny, N. P. Odgers, and A. B. McBratney
2014. Using model averaging to combine soil property rasters from legacy soil maps and from point data. *Geoderma*, 232-234:34–44.