Digital soil assessment: A simple enterprise suitability example.

Soil Security Laboratory

2018

Digital soil assessment goes beyond the goals of digital soil mapping. Digital soil assessment (DSA) can be defined (from McBratney et al. (2012)) as the translation of digital soil mapping outputs into decision making aids that are framed by the particular, contextual human-value system which addresses the question/s at hand. The concept of DSA was first framed by Carre et al. (2007) as a mechanism for assessing soil threats, assessing soil functions and for soil mechanistic simulations to assess risk based scenarios to complement policy development. Very simply DSA can be likened to the quantitative modeling of difficult-to-measure soil and attributes. An obvious candidate application for DSA is land suitability evaluation for a specified land use type, which van Diepen et al. (1991) define as all methods to explain or predict the use potential of land.

Land evaluation in some sense has been in practice at least since the earliest known human civilizations. The shift to sedentary agriculture from nomadic lifestyles is at least indicative of a concerted effort of human investment to evaluate the potential and capacity of land and its soils to support some form of agriculture like cropping (Brevik and Hartemink, 2010). In the modern times there is a well-documented history of land evaluation practice and programs throughout the world, many of which are described in Mueller et al. (2010). Much of the current thinking around land evaluation for agriculture are well documented within the land evaluation guidelines prepared by the Food and Agriculture Organization of the United Nations (FAO) in 1976 (FAO, 1976). These guidelines have strongly influenced and continue to guide land evaluation projects throughout the world. The FAO framework is a crop specific LSA system with a 5-class ranking of suitability (FAO Land Suitability Classes) from 1: Highly Suitable to 5: Permanently Not Suitable. Given a suite of biophysical information from a site, each attribute is evaluated against some expert-defined thresholds for each suitability class. The final evaluation of suitability for the site is the one in which is most limiting.

Digital soil mapping complementing land evaluation assessment is being more regularly observed. Examples (in Australia) include Kidd et al. (2012) in Tasmania and Harms et al. (2015) in Queensland. Perhaps an obvious reason is that one can derive with digital soil and climate modeling, very attribute specific mapping which can be targeted specifically to a particular agricultural land use or even to a specific enterprise (Kidd et al., 2012).

In this chapter, an example is given of a DSA where enterprise suitability is assessed. The specific example is a digital land suitability assessment (LSA) for hazelnuts across an area of northern Tasmania, Australia (Meander Valley) which has been previously described in Malone et al. (2015). For context, the digital soil assessment example has been one function of the Tasmanian Wealth from Water project for developing detailed land suitability assessments (20 specific agricultural enterprises) to support irrigated agricultural expansion across the state ((Kidd et al., 2012); (Kidd et al., 2015)). The project was commissioned for a couple of targeted areas, but has since been rolled out across the state (Kidd et al., 2015). Further general information about the project can be found at `http://dpipwe.tas.gov.au/agriculture/investing-in-irrigation/enterprise-suitability-toolkit`.

The example considered in this chapter is just to give an overview of how to perform a DSA in what could be considered as a relatively simple example. Using the most-limiting factor approach of land suitability assessment, the procedure requires a pixel-by-pixel assessment of a number of input variables which have been expertly defined as being important for the establishment and growth of hazelnuts. Malone et al. (2015) describes the digital mapping processes that went into creating the input variables for this example. The approach also assumes that the predicted maps of the input variables are also error free. Figure 1 shows an example of the input variable requirements and the suitability thresholds for hazelnuts. You will notice the biophysical variables include both soil and climatic variables, and the suitability classification has four levels of grading.

Probably the first thing to consider for enabling the DSA in this example is to codify the information in Figure 1 into an `R` function. It would look like something similar to the following script.

```
# HAZELNUT SUITABILITY ASSESSMENT FUNCTION
hazelnutSuits <- function(samp.matrix) {
    out.matrix <- matrix(NA, nrow = nrow(samp.matrix), ncol = 10)

    # Chill Hours
    out.matrix[which(samp.matrix[, 1] > 1200), 1] <- 1
    out.matrix[which(samp.matrix[, 1] > 600 & samp.matrix[, 1] <= 1200), 1] <- 2
    out.matrix[which(samp.matrix[, 1] <= 600), 1] <- 4

    # Clay content
    out.matrix[which(samp.matrix[, 2] > 10 & samp.matrix[, 2] <= 30), 2] <- 1
    out.matrix[which(samp.matrix[, 2] > 30 & samp.matrix[, 2] <= 50), 2] <- 2
    out.matrix[which(samp.matrix[, 2] > 50 | samp.matrix[, 2] <= 10), 2] <- 4

    # Soil Drainage
    out.matrix[which(samp.matrix[, 3] > 3.5), 3] <- 1
    out.matrix[which(samp.matrix[, 3] <= 3.5 & samp.matrix[, 3] > 2.5), 3] <- 2
    out.matrix[which(samp.matrix[, 3] <= 2.5 & samp.matrix[, 3] > 1.5), 3] <- 3
    out.matrix[which(samp.matrix[, 3] <= 1.5), 3] <- 1

    # EC (transformed variable)
```

Crop	Soil Depth	Depth to sodic layer	pH of top 15cm (H2O)	EC (top 15 cm)	Texture (top 15cm - % clay))	Drainage	Stoniness (top 15cm)	Frost	Mean max monthly temp	rainfall	Chill hours
W	>50cm		6.5	<0.15dS/m	10–30%	Well, Moderately well	<10% <=2 (>200mm)	No days < -6 deg C in June, July or Aug – occurs 4/5 years	Mean Jan or Feb max temp – 20-30°C	<50mm (mean March)	Chill hours 0-7° C (April-August inclusive): >1200
S	40-50cm		5.5-6.5	<0.15dS/m	30-50%	Imperfect	10-20% 3 (>200mm)	No days < -6 deg C in June, July or Aug – occurs 3/5 to 4/5 years	Mean Jan or Feb max temp – 30-33°C & 18-20°C	<50mm (mean March)	Chill hours 0-7° C (April-August inclusive): 600 - 1200
MS	30-40cm		6.5-7.1	<0.15dS/m	30-50%	Imperfect	10-20% 4 (>200mm)	No days < -6 deg C in June, July or Aug – occurs 2/5 to 3/5 years	Mean Jan or Feb max temp – 33-35°C	<50mm (mean March)	Chill hours 0-7° C (April-August inclusive): 600 - 1200
U	<30cm		<5.5 >7.1	>0.15dS/m	>50% or <10%	Poor, Very poor	>20% >=4 (>200mm)	No days < -6 deg C in June, July or Aug – occurs <2/5 years	Mean Jan or Feb max temp – >35°C & <18°C	>50mm (mean March)	Chill hours 0-7° C (April-August inclusive): <600

Well suited (W): Land having no significant limitations to sustained application of a given use, or only minor limitations that will not significantly reduce productivity or benefits and will not raise inputs above an acceptable level. Any risk of crop loss is inherently low or can be easily overcome with management practices that are easy and cheap to implement.

Suitable (S): Land having limitations which are moderately severe for sustained application of a given use; the limitations will reduce productivity or benefits and increase required inputs to the extent that the overall advantage to be gained from the use, although still attractive, will be appreciably inferior to that expected on Class S1 land. Risk of crop loss is moderately high or requires management practices that are difficult or costly to implement.

Marginally Suitable (MS): Land having limitations which are severe for sustained application of a given use and will so reduce productivity or benefits, or increase required inputs, that this expenditure will be only marginally justified. Risk of crop loss may be high.

Unsuitable (U): Land which has qualities that appear to preclude sustained use of the kind under consideration

Figure 1: Suitability parameters and thresholds for hazelnuts. Sourced from DPIPWE (2015).

```
out.matrix[which(samp.matrix[, 4] <= 0.15), 4] <- 1
out.matrix[which(samp.matrix[, 4] > 0.15), 4] <- 4

# Frost
```

```
    out.matrix[which(samp.matrix[, 10] == 0), 5] <- 1
    out.matrix[which(samp.matrix[, 10] != 0 & samp.matrix[, 5] >= 80), 5] <- 1
    out.matrix[which(samp.matrix[, 10] != 0 & samp.matrix[, 5] < 80 & samp.matrix[,
        5] >= 60), 5] <- 2
    out.matrix[which(samp.matrix[, 10] != 0 & samp.matrix[, 5] < 60 & samp.matrix[,
        5] >= 40), 5] <- 3
    out.matrix[which(samp.matrix[, 10] != 0 & samp.matrix[, 5] < 40), 5] <- 3

    # pH
    out.matrix[which(samp.matrix[, 6] <= 6.5 & samp.matrix[, 6] >= 5.5), 6] <- 1
    out.matrix[which(samp.matrix[, 6] > 6.5 & samp.matrix[, 6] <= 7.1), 6] <- 3
    out.matrix[which(samp.matrix[, 6] < 5.5 | samp.matrix[, 6] > 7.1), 6] <- 4

    # rainfall
    out.matrix[which(samp.matrix[, 7] <= 50), 7] <- 1
    out.matrix[which(samp.matrix[, 7] > 50), 7] <- 4


    # soil depth
    out.matrix[which(samp.matrix[, 13] == 0), 8] <- 1
    out.matrix[which(samp.matrix[, 13] != 0 & samp.matrix[, 8] > 50), 8] <- 1
    out.matrix[which(samp.matrix[, 13] != 0 & samp.matrix[, 8] <= 50 & samp.matrix[,
        8] > 40), 8] <- 2
    out.matrix[which(samp.matrix[, 13] != 0 & samp.matrix[, 8] <= 40 & samp.matrix[,
        8] > 30), 8] <- 3
    out.matrix[which(samp.matrix[, 13] != 0 & samp.matrix[, 8] <= 30), 8] <- 4

    # temperature
    out.matrix[which(samp.matrix[, 9] > 20 & samp.matrix[, 9] <= 30), 9] <- 1
    out.matrix[which(samp.matrix[, 9] > 30 & samp.matrix[, 9] <= 33 | samp.matrix[,
        9] <= 20 & samp.matrix[, 9] > 18), 9] <- 2
    out.matrix[which(samp.matrix[, 9] > 33 & samp.matrix[, 9] <= 35), 9] <- 3
    out.matrix[which(samp.matrix[, 9] > 35 | samp.matrix[, 9] <= 18), 9] <- 4

    # rocks
    out.matrix[which(samp.matrix[, 11] == 0), 10] <- 1
    out.matrix[which(samp.matrix[, 11] != 0 & samp.matrix[, 12] <= 2), 10] <- 1
    out.matrix[which(samp.matrix[, 11] != 0 & samp.matrix[, 12] == 3), 10] <- 2
    out.matrix[which(samp.matrix[, 11] != 0 & samp.matrix[, 12] == 4), 10] <- 3
    out.matrix[which(samp.matrix[, 11] != 0 & samp.matrix[, 12] > 4), 10] <- 4
    return(out.matrix)
}
```

Essentially the function takes in a matrix of *(*n) number of rows by 13 columns. The number of columns is fixed as each coincides with one of the biophysical variables. There are some variables where there is relevant information contained in two columns. For example, columns assigned to soil depth information are contained in columns 8 and 13. The reason is such that soil depth is modeled via a two-stage process (as is exemplified in the chapter

regarding two-stage modeling for DSM), such that a model is used to predict the presence/absence of a certain condition —which for soil depth is whether lithic contact is achieved less than 1.5 m from the soil surface — followed by a secondary model that predicts soil depth (where soil is expected to be less than 1.5m). Subsequently the secondary model is only invoked if there is a positive condition found for the first model. As such the above `R` function provides a good example of the use of sub-setting and applying conditional queries within `R`. The function `hazelnutSuits` will return a new matrix with n rows and 10 columns. The entries for each column and row will be the suitability assessment for each biophysical variable. It is then just a matter of determining what the maximum value is on each row in order to give an overall suitability valuation (as this is the most limiting factor approach). This can be achieved using the `rowMaxs` function from the `matrixStats` package.

```
# rowMaxs(output from hazelnutSuits function)
```

Following is a workflow for implementing the `hazelnutSuits` function, with special application of it in a spatial mapping context.

1 Mapping example of digital land suitability assessment

Assuming there are digital soil and climate maps already created for use in the hazelnut land suitability assessment, it is relatively straightforward to run the LSA. Keep in mind that the creation of the biophysical variable maps were created via a number of means which included continuous attribute modeling, binomial and ordinal logistic regression, and a combination of both i.e. through the two-stage mapping process. So let's get some sense of the LSA input variables.

```
# LSA input variables
library(raster)
names(lsa.variables)
```

```
##  [1] "X1_chill_HAZEL_FINAL_meander"
##  [2] "X2_clay_FINAL_meander"
##  [3] "X3_drain_FINAL_meander"
##  [4] "X4_EC_cubist_meander"
##  [5] "X5_Frost_HAZEL_FINAL_meander"
##  [6] "X6_pH_FINAL_meander"
##  [7] "X7_rain_HAZEL_FINAL_meander"
##  [8] "X8_soilDepth_FINAL_meander"
##  [9] "X9_temp_HAZEL_FINAL_meander"
## [10] "X5_Frost_HAZEL_binaryClass_meander"
## [11] "X10_rocks_binaryClass_meander"
## [12] "X11_rocks_ordinalClass_meander"
## [13] "X8_soilDepth_binaryClass_meander"
```

```
class(lsa.variables)
```

```
## [1] "RasterStack"
## attr(,"package")
## [1] "raster"
```

```
# Raster stack dimensions
dim(lsa.variables)
```

```
## [1] 1081 1685   13
```

```
# Raster resolution
res(lsa.variables)
```

```
## [1] 30 30
```

So there are 13 rasters of data, which you will note coincide with the inputs required for the `hazelnutSuits` function. Now all that is required is to go pixel by pixel and apply the LSA function. In `R` the implementation may take the following form:

```
# Retrieve values of from all rasters for a given row Here it is the 1st row
# of the raster
cov.Frame <- getValues(lsa.variables, 1)
nrow(cov.Frame)
```

```
## [1] 1685
```

```
# Remove the pixels where there is NA or no data present
sub.frame <- cov.Frame[which(complete.cases(cov.Frame)), ]
nrow(sub.frame)
```

```
## [1] 27
```

```
names(sub.frame)
```

```
## NULL
```

```
# run hazelnutSuits function
hazel.lsa <- hazelnutSuits(sub.frame)

# Assess for suitability
library(matrixStats)
rowMaxs(hazel.lsa)
```

```
##  [1] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 4 4 4 4 4 4 4
```

The above script has just performed the hazelnut LSA upon the entire first row of the input variable rasters. There were only 27 pixels where the full suite of data was available in this case. To do the LSA for the entire mapping extent we could effectively run the script above for each row of the input rasters. Naturally, this would take an age to do manually, so it might be more appropriate to use the script above inside a `for` loop where the row index changes for each loop. Alternatively, the custom `hazelnutSuits` function could be used as an input argument for the `raster` package `calc` function, or better still using the `clusterR` function if there is a need to do the LSA in parallel mode across multiple compute nodes. Given the available options, we will demonstrate the mapping process using the looping approach. While it

may be computationally slower, it imprints the concept of applying the LSA spatially with greater clarity.

From the above script and resulting output, we essentially returned a vector of integers. We effectively lost all links to the fact that the inputs and resulting outputs are spatial data. Subsequently there is a need to link the suitability assessment back to the mapping. Fortunately we know the column positions of the instances where there was the full suite of input data. So once we have set up a raster object to which data can be written to (which has the same raster properties of the input data), it is just a matter of placing the LSA outputs into the row and column positions as those of the input data. A full example is given below, but for the present purpose, the LSA output data placement into a raster would look something like the following:

```
# A one column matrix with number of rows equal to number of columns in
# raser inputs
a.matrix <- matrix(NA, nrow = nrow(cov.Frame), ncol = 1)

# Place LSA outputs into correct row positions
a.matrix[which(complete.cases(cov.Frame)), 1] <- rowMaxs(hazel.lsa)

# Write LSA outputs to raster object (1st row)
LSA.raster <- writeValues(LSA.raster, a.matrix[, 1], 1)
```

Naturally the above few lines of script would also be embedded into the looping process as described above. Below is an example of putting all these operations together to ultimately produce a map of the suitability assessment. To add a slight layer of complexity, we may also want to produce maps of the suitability assessment for each of the input variables. This helps in determining which factors are causing the greatest limitations and where they occur. First, we need to create a number of rasters to which we can write the outputs of the LSA to.

```
# Create a suite of rasters of same raster properties is LSA input variables
# Overall suitability classification
LSA.raster <- raster(lsa.variables[[1]])

# Write outputs of LSA directly to file
LSA.raster <- writeStart(LSA.raster, filename = "meander_MLcat.tif", format = "GTiff",
    dataType = "INT1S", overwrite = TRUE)


# Individual LSA input suitability rasters
mlf1 <- raster(lsa.variables[[1]])
mlf2 <- raster(lsa.variables[[1]])
mlf3 <- raster(lsa.variables[[1]])
mlf4 <- raster(lsa.variables[[1]])
mlf5 <- raster(lsa.variables[[1]])
mlf6 <- raster(lsa.variables[[1]])
mlf7 <- raster(lsa.variables[[1]])
mlf8 <- raster(lsa.variables[[1]])
```

```
mlf9 <- raster(lsa.variables[[1]])
mlf10 <- raster(lsa.variables[[1]])

# Also write LSA outputs directly to file.
mlf1 <- writeStart(mlf1, filename = "meander_Haz_mlf1_CAT.tif", format = "GTiff",
    dataType = "INT1S", overwrite = TRUE)
mlf2 <- writeStart(mlf2, filename = "meander_Haz_mlf2_CAT.tif", format = "GTiff",
    dataType = "INT1S", overwrite = TRUE)
mlf3 <- writeStart(mlf3, filename = "meander_Haz_mlf3_CAT.tif", format = "GTiff",
    dataType = "INT1S", overwrite = TRUE)
mlf4 <- writeStart(mlf4, filename = "meander_Haz_mlf4_CAT.tif", format = "GTiff",
    dataType = "INT1S", overwrite = TRUE)
mlf5 <- writeStart(mlf5, filename = "meander_Haz_mlf5_CAT.tif", format = "GTiff",
    dataType = "INT1S", overwrite = TRUE)
mlf6 <- writeStart(mlf6, filename = "meander_Haz_mlf6_CAT.tif", format = "GTiff",
    dataType = "INT1S", overwrite = TRUE)
mlf7 <- writeStart(mlf7, filename = "meander_Haz_mlf7_CAT.tif", format = "GTiff",
    dataType = "INT1S", overwrite = TRUE)
mlf8 <- writeStart(mlf8, filename = "meander_Haz_mlf8_CAT.tif", format = "GTiff",
    dataType = "INT1S", overwrite = TRUE)
mlf9 <- writeStart(mlf9, filename = "meander_Haz_mlf9_CAT.tif", format = "GTiff",
    dataType = "INT1S", overwrite = TRUE)
mlf10 <- writeStart(mlf10, filename = "meander_Haz_mlf10_CAT.tif", format = "GTiff",
    dataType = "INT1S", overwrite = TRUE)
```

Now we can implement the `for` loop procedure and do the LSA for the entire mapping extent.

```
# Run the suitability model: Open loop:for each row of each input raster get
# raster values
for (i in 1:dim(LSA.raster)[1]) {
    cov.Frame <- getValues(lsa.variables, i)
    # get the complete cases
    sub.frame <- cov.Frame[which(complete.cases(cov.Frame)), ]

    # Run hazelnut LSA function
    t1 <- hazelnutSuits(sub.frame)

    # Save results to raster
    a.matrix <- matrix(NA, nrow = nrow(cov.Frame), ncol = 1)
    a.matrix[which(complete.cases(cov.Frame)), 1] <- rowMaxs(t1)
    LSA.raster <- writeValues(LSA.raster, a.matrix[, 1], i)


    # Also save the single input variable assessment outputs
    mlf.out <- matrix(NA, nrow = nrow(cov.Frame), ncol = 10)
    mlf.out[which(complete.cases(cov.Frame)), 1] <- t1[, 1]
    mlf.out[which(complete.cases(cov.Frame)), 2] <- t1[, 2]
    mlf.out[which(complete.cases(cov.Frame)), 3] <- t1[, 3]
    mlf.out[which(complete.cases(cov.Frame)), 4] <- t1[, 4]
```

```
    mlf.out[which(complete.cases(cov.Frame)), 5] <- t1[, 5]
    mlf.out[which(complete.cases(cov.Frame)), 6] <- t1[, 6]
    mlf.out[which(complete.cases(cov.Frame)), 7] <- t1[, 7]
    mlf.out[which(complete.cases(cov.Frame)), 8] <- t1[, 8]
    mlf.out[which(complete.cases(cov.Frame)), 9] <- t1[, 9]
    mlf.out[which(complete.cases(cov.Frame)), 10] <- t1[, 10]
    mlf1 <- writeValues(mlf1, mlf.out[, 1], i)
    mlf2 <- writeValues(mlf2, mlf.out[, 2], i)
    mlf3 <- writeValues(mlf3, mlf.out[, 3], i)
    mlf4 <- writeValues(mlf4, mlf.out[, 4], i)
    mlf5 <- writeValues(mlf5, mlf.out[, 5], i)
    mlf6 <- writeValues(mlf6, mlf.out[, 6], i)
    mlf7 <- writeValues(mlf7, mlf.out[, 7], i)
    mlf8 <- writeValues(mlf8, mlf.out[, 8], i)
    mlf9 <- writeValues(mlf9, mlf.out[, 9], i)
    mlf10 <- writeValues(mlf10, mlf.out[, 10], i)
    print((dim(LSA.raster)[1]) - i)
}  #END OF LOOP

# complete writing rasters to file
LSA.raster <- writeStop(LSA.raster)
mlf1 <- writeStop(mlf1)
mlf2 <- writeStop(mlf2)
mlf3 <- writeStop(mlf3)
mlf4 <- writeStop(mlf4)
mlf5 <- writeStop(mlf5)
mlf6 <- writeStop(mlf6)
mlf7 <- writeStop(mlf7)
mlf8 <- writeStop(mlf8)
mlf9 <- writeStop(mlf9)
mlf10 <- writeStop(mlf10)
```

As you may encounter, the above script can take quite a while to complete, but ultimately you should be able to produce a number of mapping products. Figure 2 shows the map of the overall suitability classification and the script to produce it is below.

```
library(rasterVis)

LSA.raster <- as.factor(LSA.raster)
rat <- levels(LSA.raster)[[1]]
rat[["suitability"]] <- c("Well Suited", "Suited", "Moderately Suited", "Unsuited")
levels(LSA.raster) <- rat

# plot
area_colors <- c("#FFFF00", "#1D0BE0", "#1CEB15", "#C91601")
levelplot(LSA.raster, col.regions = area_colors, xlab = "", ylab = "")
```

Similarly the above plotting procedure can be repeated to look at single input variable limitations too.

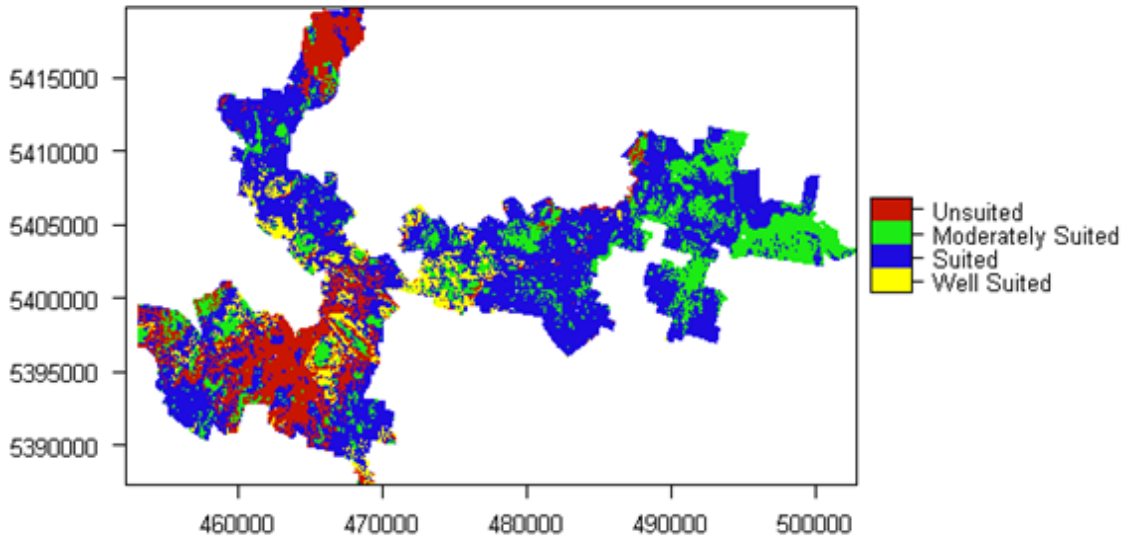

Figure 2: Digital suitability assessment for hazelnuts across the Meander Valley, Tasmania (assuming all LSA input variables are error free).

The approaches detailed in this chapter are described in greater detail in Malone et al. (2015) within the context of taking account of uncertainties within LSA. Taking account of the input variable uncertainties adds an additional level of complexity to what was achieved above, but is an important consideration nonetheless, as the resulting outputs can be assessed for reliability in an objective manner. However, that particular workflow for LSA is not covered in this chapter as it is only meant to provide a general perspective and relatively simple example of a real world digital soil assessment.

References

Brevik, E. C. and A. E. Hartemink
2010. Early soil knowledge and the birth and development of soil science. *Catena*, 83(1):23 – 33.

Carre, F., A. B. McBratney, T. Mayr, and L. Montanarella
2007. Digital soil assessments: Beyond dsm. *Geoderma*, 142(1-2):69 – 79.

DPIPWE
2015. Enterprise suitability toolkit [online] dpipwe.tas.gov.au.

FAO
1976. *A Framework for Land Evaluation. Soils Bulletin 32.* Rome: Food and Agriculture Organisation of the United Nations.

Harms, B., D. Brough, S. Philip, R. Bartley, D. Clifford, M. Thomas,

R. Willis, and L. Gregory
2015. Digital soil assessment for regional agricultural land evaluation. *Global Food Security*, 5:25 – 36. Special Section on 3rd.

Kidd, D., M. Webb, B. Malone, B. Minasny, and A. McBratney
2015. Digital soil assessment of agricultural suitability, versatility and capital in tasmania, australia. *Geoderma Regional*, 6:7 – 21.

Kidd, D. B., M. A. Webb, C. J. Grose, R. M. Moreton, B. P. Malone, A. B. McBratney, B. Minasny, R. Viscarra-Rossel, L. A. Sparrow, and R. Smith
2012. Digital soil assessment: Guiding irrigation expansion in tasmania, australia. In *Digital Soil Assessment and Beyond*, B. Minasny, B. P. Malone, and A. B. McBratney, eds., Pp. 3–9, Boca Raton. CRC.

Malone, B. P., D. B. Kidd, B. Minasny, and A. B. McBratney
2015. Taking account of uncertainties in digital land suitability assessment. *PeerJ*, P. 3:e1366.

McBratney, A. B., B. Minasny, I. Wheeler, B. P. Malone, and D. V. D. Linden
2012. Frameworks for digital soil assessment. In *Digital Soil Assessments and Beyond*, B. Minasny, B. P. Malone, and A. B. McBratney, eds., Pp. 9–15, London, UK. CRC Press.

Mueller, L., U. Schindler, W. Mirschel, T. Shepherd, B. Ball, K. Helming, J. Rogasik, F. Eulenstein, and H. Wiggering
2010. Assessing the productivity function of soils. a review. *Agronomy for Sustainable Development*, 30(3):601–614.

van Diepen, C., H. van Keulen, J. Wolf, and J. Berkhout
1991. Land evaluation: From intuition to quantification. In *Advances in Soil Science*, B. Stewart, ed., volume 15 of *Advances in Soil Science*, Pp. 139–204. Springer New York.