Categorical soil attribute modeling and mapping: Multinomial logisitc regression.

Soil Security Laboratory

2018

1 Multinomial logistic regression

Multinomial logistic regression is used to model nominal outcome variables, in which the log odds of the outcomes are modeled as a linear combination of the predictor variables. Because we are dealing with categorical variables, it is necessary that logistic regression take the natural logarithm of the odds (log-odds) to create a continuous criterion. The logit of success is then fit to the predictors using regression analysis. The results of the logit, however are not intuitive, so the logit is converted back to the odds via the inverse of the natural logarithm, namely the exponential function. Therefore, although the observed variables in logistic regression are categorical, the predicted scores are modeled as a continuous variable (the logit). The logit is referred to as the link function in logistic regression. As such, although the output in logistic regression will be multinomial, the logit is an underlying continuous criterion upon which linear regression is conducted. This means that for logistic regression we are able to return the most likely or probable prediction (class) as well as the probabilities of occurrence for all the other classes considered. Some discussion of the theoretical underpinnings of multinomial logistic regression, and importantly its application in DSM is given in Kempen et al. (2009).

In `R` we can use the `multinom` function from the `nnet` package to perform logistic regression. The are other implementations of this model in `R`, so it is worth a look to compare and contrast them. Fitting `multinom` is just like fitting a linear model as seen below.

As described earlier, the data to be used for the following modelling exercises are Terron classes as sampled from the map presented in Malone et al. (2014).The sample data contains 1000 entries of which there are 12 different Terron classes. Before getting into the modeling, we first load in the data and then perform the covariate layer intersection using the suite of environmental variables contained in the `hunterCovariates` data object in the `ithir` package. The small selection of covariates cover an area of approximately $220km^2$ at a spatial resolution of 25m. They include those derived from a DEM: altitude above channel network (AACN), solar light insolation and terrain wetness index (TWI). Gamma radiometric data (total count) is also included together with a surface that depicts soil drainage in form of a continuous index (ranging from 0 to 5). These 5 different covariate layers are stacked together via a `rasterStack`.

```
library(ithir)
library(sp)
library(raster)

data(hvTerronDat)
data(hunterCovariates)
```

Transform the `hvTerronDat` data to a `SpatialPointsDataFrame`.

```
names(hvTerronDat)
```

```
## [1] "x"      "y"      "terron"
```

```
coordinates(hvTerronDat) <- ~x + y
```

As these data are of the same spatial projection as the `hunterCovariates`, there is no need to perform a coordinate transformation. So we can perform the intersection immediately.

```
DSM_data <- extract(hunterCovariates, hvTerronDat, sp = 1, method = "simple")
DSM_data <- as.data.frame(DSM_data)
str(DSM_data)
```

```
## 'data.frame': 1000 obs. of  8 variables:
##  $ x                : num  346535 334760 340910 336460 344510 ...
##  $ y                : num  6371941 6375841 6377691 6382041 6378116 ...
##  $ terron           : Factor w/ 12 levels "1","2","3","4",..: 3 4 12 5 6 11 3 10 3 5 ..
##  $ AACN             : num  37.544 25.564 32.865 0.605 9.516 ...
##  $ Drainage.Index   : num  4.72 4.78 2 4.19 4.68 ...
##  $ Light.Insolation : num  1690 1736 1712 1712 1677 ...
##  $ TWI              : num  11.5 13.8 13.4 18.6 19.8 ...
##  $ Gamma.Total.Count: num  380 407 384 388 454 ...
```

It is always good practice to check to see if any of the observational data returned any NA values for any one of the covariates. If there is NA values, it indicates that the observational data is outside the extent of the covariate layers. It is best to remove these observations from the data set.

```
which(!complete.cases(DSM_data))
```

```
## integer(0)
```

```
DSM_data <- DSM_data[complete.cases(DSM_data), ]
```

Now for model fitting. The target variable is `terron`. So lets just use all the available covariates in the model. We will also subset the data for an external validation i.e. random hold back validation.

```
library(nnet)

set.seed(655)
training <- sample(nrow(DSM_data), 0.7 * nrow(DSM_data))
hv.MNLR <- multinom(terron ~ AACN + Drainage.Index + Light.Insolation + TWI +
    Gamma.Total.Count, data = DSM_data[training, ])
```

```
## # weights:  84 (66 variable)
## initial  value 1739.434655
## iter  10 value 1544.694562
## iter  20 value 1368.990795
## iter  30 value 1236.748119
## iter  40 value 1161.299395
## iter  50 value 1063.598067
## iter  60 value 992.924808
## iter  70 value 968.797316
## iter  80 value 963.189954
## iter  90 value 962.000276
## iter 100 value 961.634415
## final  value 961.634415
## stopped after 100 iterations
```

Using the `summary` function allows us to see the linear models for each Terron class, which are the result of the log-odds of each soil class modeled as a linear combination of the covariates. We can also see the probabilities of occurrence for each Terron class at each observation location by using the `fitted` function.

```
summary(hv.MNLR)

# Estimate class probabilities
probs.hv.MNLR <- fitted(hv.MNLR)

# return top of data frame of probabilites
head(probs.hv.MNLR)
```

Subsequently, we can also determine the most probable Terron class using the the `predict` function.

```
pred.hv.MNLR <- predict(hv.MNLR)
summary(pred.hv.MNLR)
```

```
##   1   2   3   4   5   6   7   8   9  10  11  12
##  21   7  60  62 110  73 169 115  18  29  12  24
```

Lets now perform an internal validation of the model to assess its general performance. Here we use the `goofcat` function, but this time we import the two vectors into the function which correspond to the observations and predictions respectively.

```
goofcat(observed = DSM_data$terron[training], predicted = pred.hv.MNLR)
```

```
## $confusion_matrix
##       1 2  3  4  5  6  7  8  9 10 11 12
## 1    13 3  1  2  0  0  1  0  1  0  0  0
## 2     1 5  0  0  0  0  0  0  1  0  0  0
## 3     0 1 35  2  0  0  8  0  2  3  6  3
## 4     4 0  9 30  1  1  3  3  5  0  6  0
## 5   0 0  0  0 56 10  1 10 13 17  1  2
## 6   0 0  0  1 16 47  0  6  1  2  0  0
## 7   0 0 14  7  4  0 92 16  7  8  9 12
## 8   0 0  0  4 10  7 14 57  4  6 13  0
## 9   0 0  0  5  3  0  2  2  5  0  1  0
## 10  0 0  0  0  4  1  2  6  3 13  0  0
## 11  0 0  4  3  0  0  0  3  0  0  2  0
## 12  1 0  2  1  0  0  1  0  0  4  0 15
##
## $overall_accuracy
## [1] 53
##
## $producers_accuracy
##  1  2  3  4  5  6  7  8  9 10 11 12
## 69 56 54 55 60 72 75 56 12 25  6 47
##
## $users_accuracy
##  1  2  3  4  5  6  7  8  9 10 11 12
## 62 72 59 49 51 65 55 50 28 45 17 63
##
## $kappa
## [1] 0.4637285
```

Similarly, performing the external validation requires first using the `pred.hv.MNLR` model to predict on the withheld points.

```
V.pred.hv.MNLR <- predict(hv.MNLR, newdata = DSM_data[-training, ])
goofcat(observed = DSM_data$terron[-training], predicted = V.pred.hv.MNLR)
```

```
## $confusion_matrix
##    1 2  3 4  5  6  7  8 9 10 11 12
## 1  1 1  1 2  0  0  0  0 1  0  0  0
## 2  5 0  2 0  0  0  0  0 1  1  0  0
## 3  0 0 13 1  0  0  5  0 1  2  2  2
## 4  2 3  9 8  0  0  0  4 8  0  3  0
## 5  0 0  0 0 21  7  0  8 6  3  0  2
## 6  0 0  0 1  8 18  0  5 1  0  0  0
## 7  0 0  8 4  0  0 38  9 4  5  2  5
## 8  0 0  0 3  3  0  6 15 1  3  1  0
## 9  0 0  0 1  2  0  0  1 0  0  0  0
## 10 0 0  0 0  4  1  1  4 1  9  2  0
## 11 0 0  0 0  0  0  0  1 0  0  0  0
## 12 0 0  1 0  0  0  1  0 0  2  0  4
##
## $overall_accuracy
## [1] 43
##
## $producers_accuracy
##  1  2  3  4  5  6  7  8  9 10 11 12
## 13  0 39 40 56 70 75 32  0 36  0 31
```

```
##
## $users_accuracy
##  1  2  3  4  5  6  7  8  9 10 11 12
## 17  0 50 22 45 55 51 47  0 41  0 50
##
## $kappa
## [1] 0.3476539
```

Using the raster `predict` function is the method for applying the `hv.MNLR` model across the whole area. Note that the `clusterR` function can also be used here too if there is a requirement to perform the spatial prediction across multiple compute nodes. Note also that it is also possible for multinomial logistic regression to create the map of the most probable class, as well as the probabilities for all classes. The first script example below is for mapping the most probable class which is specified by setting the `type` parameter to "`class`". If probabilities are required "`probs`" would be used for the `type` parameter, together with specifying an `index` integer to indicate which class probabilities you wish to map. The second script example below shows the parametisation for predicting the probabilities for Terron class 1.

```
# class prediction
map.MNLR.c <- predict(hunterCovariates, hv.MNLR, type = "class",
filename = "hv_MNLR_class.tif",
    format = "GTiff", overwrite = T, datatype = "INT2S")

# class probabilities
map.MNLR.p <- predict(hunterCovariates, hv.MNLR, type = "probs", index = 1,
    filename = "edge_MNLR_probs1.tif", format = "GTiff", overwrite = T,
datatype = "FLT4S")
```

Plotting the resulting class map is not as straightforward as for mapping continuous variables. A solution is scripted below which uses an associated package to `raster` called `rasterVis`. You will also note the use of explicit colors for each Terron class as they were the same colors used in the Terron map presented by Malone et al. (2014). The colors are defined in terms of HEX color codes. A very good resource for selecting colors or deciding on color ramps for maps is *colorbrewer* located at `http://colorbrewer2.org/`. The script below produces the map in Figure 1.

```
map.MNLR.c <- as.factor(map.MNLR.c)

## Add a land class column to the Raster Attribute Table
rat <- levels(map.MNLR.c)[[1]]
rat[["terron"]] <- c("HVT_001", "HVT_002", "HVT_003", "HVT_004", "HVT_005",
    "HVT_006", "HVT_007", "HVT_008", "HVT_009", "HVT_010", "HVT_011", "HVT_012")
levels(map.MNLR.c) <- rat

## HEX colors
area_colors <- c("#FF0000", "#38A800", "#73DFFF", "#FFEBAF", "#A8A800", "#0070FF",
    "#FFA77F", "#7AF5CA", "#D7B09E", "#CCCCCC", "#B4D79E", "#FFFF00")
# plot
```

```r
levelplot(map.MNLR.c, col.regions = area_colors, xlab = "", ylab = "")
```

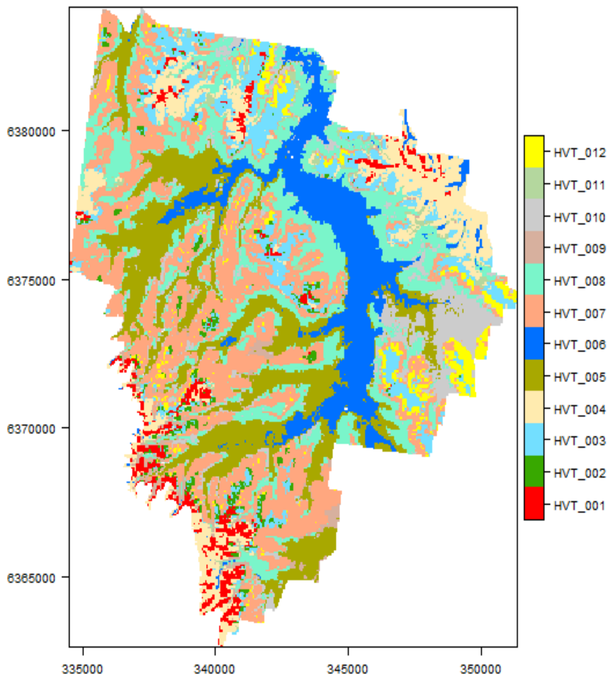

Figure 1: Hunter Valley most probable Terron class map created using multinomial logistic regression model

References

Kempen, B., D. J. Brus, G. B. M. Heuvelink, and J. J. Stoorvogel 2009. Updating the 1:50,000 dutch soil map using legacy soil data: A multinomial logistic regression approach. *Geoderma*, 151:311–326.

Malone, B. P., P. Hughes, A. B. McBratney, and B. Minsany 2014. A model for the identification of terrons in the lower hunter valley, australia. *Geoderma Regional*, 1:31–47.