Last Updated: October 2022
Software: R version 4.2.1 (or higher) and RStudio

EXERCISE 4
Introduction to Modeling & Prediction in R

Introduction

In this exercise, we will put together some of the pieces that you have learned in the previous exercises and you will learn how to run a geostatistical model to create a model prediction raster. You will fit a simple Random Forest regression model predicting total stand biomass using raster predictor layers. You will use this model to create a predicted stand biomass raster layer which may be used to map biomass.

Objectives

- Extract raster values to spatial points to assemble a modeling dataset
- Fit a simple Random Forests regression model
- Use the classification model to predict stand biomass wall-to-wall for a subset of the study area

Required Software & Data

- **R version 4.2.1 (or higher) -** Go to the RStudio CRAN mirror and follow the instructions to download and install R.
- **RStudio** – Download from Rstudio.
- **Course data folder**

Prerequisites

- Introduction to Geospatial Scripting
- Introduction to Random Forests
- Completion of Exercise 3 of this course – Raster Processing & Analysis in R

Table of Contents

Part 1: Set up your script, load packages and datasets

A. Set up a new R script for this exercise

1. From the file menu, open a new R script, save it in your scripts folder using the name **04_SpatialModelingAndPrediction.R**

B. Install and load the required packages:

1. Use the library function to load the terra package by adding and running the code below

```
#Load packages
library(terra)
```

2. Install and load the randomForest package for modeling

    i. Add and run the code below

```
install.packages('randomForest')
library(randomForest)
```

C. Set the workspace

1. For convenience, we identify the workspace so that we easily read from the course data folder.

    i. Add the code below to your script

```
#identify the workspace, the course data folder
workspace <-
"D:/2022/05_GeospatialScriptingInR/Data/DataForGeospatialScriptinginR/"
```

D. Load all the datasets for this exercise

1. Load the forest plot point dataset

```
#Read in shapefile – Recall that arguments are: workspace, path to
shapefile + extension
NK_points <- terra::vect(paste0(workspace,
"vector/NorthKaibabForestInvPlots.shp"))
```

2. Load a prepared raster stack that has been saved as a .tif file. This stack includes the raster predictor layers that you have explored and created in the previous exercises. They have been preprocessed and stacked for convenience.

    i. Add and run the code below.

```
# Load a stack of predictor layer rasters
predictorStack <- rast(paste0(workspace,
"/raster/predictionRasterStack.tif"))
```

3. Examine the contents and print the names of the layers in the raster stack

    i. Add and run the code below.

```
# Examine the stack
predictorStack
# Get the names of the layers
names(predictorStack)
```

4. One inconvenient thing about the .tif data format is that it doesn't save the layer names. We'll supply the layer names for you here.

```
#re-assign names to the predictor stack
names(predictorStack) <-
c("Slope","CanopyHeight","CanopyDensity","NDVI_trend2001_2016")
```

Note that the predictor layer raster stack has been put together and saved as a .tif file. To assemble the predictor layers in this way for modeling purposes, they must all have the same extent and resolution. This data preparation is easily accomplished using functions available in the terra package. To learn more about how to prepare raster layers to stack for modeling, see the project(), resample(), extend(), crop() and mask() functions.

Part 2: Assemble a modeling data set

Prior to model fitting, we need to spatially relate the values in the stack of raster predictor layers to the reference plot data. To do this, we will 'extract' the values to the plot point locations to assemble a modeling data frame that we can then use in our model fitting, evaluation, etc.

A. Use the extract function to get the values from the stack

1. Call the extract function in the terra package to get values from the predictorStack object at the point locations.
   i. Use help(extract) to open the documentation to get the order of arguments, etc.
   ii. Add and run the code below to create a new object called pointVals. What kind of object do you think this is?

```
# Extract values from the rasters to the point locations and make a
data frame
pointVals <- extract(predictorStack, NK_points)
# Print the object type to answer the question above
is(pointVals)
```

B. Create a new data frame combining the attribute table from the points and the raster values

1. Use the cbind function to bind the columns of the spatial point data frame to the extracted values. Create a new data frame called modelDF and set it equal to this new data frame.
   i. Add and run the code below to get the data frame table from the point dataset (see the note below) and bind the columns to the new point value dataset.

```
# Bind the extracted values to the NK forest plot data table to make a
data frame with values and metadata
modelDF <- cbind(data.frame(NK_points), pointVals)
```

2. Use the head() function and the names() function to take a look at the new data frame.

Examine the code above and note the data.frame() function. This function allows us to build a data frame from lists and vectors—and also allows us to access only the data components of a spatial object (as opposed to the spatial components). Use the str() function to print out the structure of an object. This will give you the names of all the components (attributes) in the object. Try it out by running str(NK_points)

```
#inspect
head(modelDF)
names(modelDF)
```

3. When you look at the first 6 rows of the data frame using the head() function, you can see that there's an additional column named "ID" that wasn't present in our NK_points data frame or our predictorStack. This ID column was created when we used the extract() function. Since we joined the point values back with the source data frame, we no longer need this additional column and we can set it to NULL.

```
# remove secondary ID field
modelDF$ID <- NULL
```

Part 3: Explore relationships among variables prior to modeling

In geostatistical modeling, we generally assemble predictor layers based on hypothesized relationships between the phenomena that we are modeling (response variable) and the selected predictor layers. In this section, we will use simple methods in R to explore the relationships between our response variable, stand total biomass, and the selected continuous raster predictor layers.

A. Print the names in the modeling dataset

1. Use the names() function on the dataset to print the names of the variables. This will let you use copy/paste from the console when you reference the variables in your script.

```
#inspect variables present
names(modelDF)
```

```
> names(modelDF)
 [1] "PLOT_ID"           "GUILD"          "DOM_TYPE"          "TCUFT"
 [5] "STBIOMS"           "Elev_ave"       "all_1st_co"        "logElev_av"
 [9] "logAll_1st"        "Slope"          "CanopyHeight"      "CanopyDensity"
[13] "NDVI_trend2001_2016"
```

Figure 1. Screenshot of the names in the assembled model dataset.

B. Plot the relationships

1. Use the plot function to plot several scatter plots relating stand total biomass to the predictor layers in our dataset.

   i. Copy, paste and run the code below – Which variables appear to relate to biomass?

```
#Look at relationships between variables
plot(modelDF$STBIOMS, modelDF$Slope)
plot(modelDF$STBIOMS, modelDF$CanopyDensity)
plot(modelDF$STBIOMS, modelDF$CanopyHeight)
plot(modelDF$STBIOMS, modelDF$NDVI_trend2001_2016)
```

2. We can also use an additional plotting command to include all of the plots that we just created in a single plot. This helps facilitate comparisons between these relationships.

```
# plot all 4 plots together for comparison
par(mfrow=c(2,2)) # set up a plotting window with 2 rows and 2 columns
plot(modelDF$STBIOMS, modelDF$Slope)
plot(modelDF$STBIOMS, modelDF$CanopyDensity)
plot(modelDF$STBIOMS, modelDF$CanopyHeight)
plot(modelDF$STBIOMS, modelDF$NDVI_trend2001_2016)
par(mfrow=c(1,1)) # reset plotting window to 1 row x 1 column
```

C. Calculate correlations between stand total biomass and predictor variables

1. Use the cor() function to calculate the correlations between stand total biomass and the predictor layers in the raster stack

   i. Copy, paste and run the code below – Which have the largest correlation with biomass?

```
#Look at correlations
cor(modelDF$STBIOMS , modelDF$Slope)
cor(modelDF$STBIOMS , modelDF$CanopyDensity)
cor(modelDF$STBIOMS , modelDF$CanopyHeight)
cor(modelDF$STBIOMS , modelDF$NDVI_trend2001_2016)
```

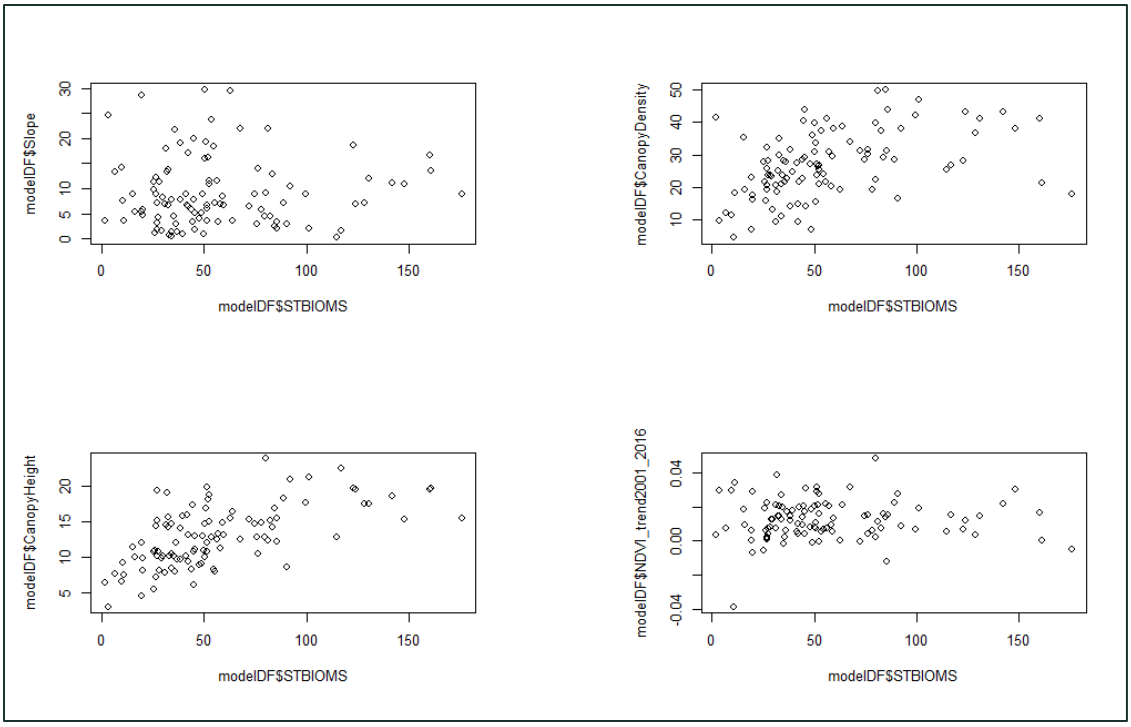

Figure 2. Scatterplot comparisons of predictor variables (y-axis) and stand total biomass (x-axis) show a visible relationship between lidar-measured canopy height and density (correlation of 0.45 and 0.62) and no real relationship between slope or the derived trend in NDVI.

Part 4: Fit stand biomass regression models

In this section, we will use the Random Forests package to fit a random forest regression model. We will look at two models, one using all the predictor variables and another just using the lidar-derived variables to predict stand total biomass. This section is not intended to provide a complete lesson on geostatistical modeling, rather simply demonstrate how R can be used to accomplish a complete modeling and prediction workflow. For more information on modeling and classification using Random Forests, check out the self-paced GTAC Random Forests tutorial.

A. Define a model predicting stand total biomass regression model

1. Create a new model object, model1, and set it equal to the model formula for predicting stand total biomass from all the variables in the raster stack.

```
#Define a model with all the variables
model1 <- STBIOMS ~ Slope + CanopyDensity + CanopyHeight +
NDVI_trend2001_2016
```

Please refer to this tutorial provided by the RSpatial Hub for a more in-depth discussion and explanation of model formulas in R.

2. Use the randomForest function to fit the model using the data in the modeling data frame.

```
rf1 <- randomForest(model1, data=modelDF)
```

B. Look at model summary and the variable importance

1. Print the model to view the summary and results
   i. Add and run the code below.

```
rf1
```

2. Use the Random Forest variable importance plot function to look at the relative importance of the predictor variables in the model prediction stand total biomass
   i. Add and run the code below.

```
varImpPlot(rf1)
```

C. Fit a simplified model and compare

1. Use the code below to define, fit and quickly evaluate a simplified model using only the most important, predicative variables

```
#Define a model with two variables
model2 <- STBIOMS ~ CanopyDensity + CanopyHeight
rf2 <- randomForest(model2, data=modelDF)
rf2
varImpPlot(rf2)
```

D. Compare this to a simple linear model

1. Add and run the code below to fit a simple linear model with the same variables

```
#Compare to a linear model
lm2 <- lm(model2, data=modelDF)
summary(lm2)
```

Which model do you think is best and why? Model fitting and selection goes well beyond the scope of this introductory R course. We can however, summarize a bit of often-cited and trusted wisdom, reminding the reader to consider Occam's razor. This principle tells us that the simpler model is usually better. In other words, the more complex a model is, the more assumptions that are made and less likely it is to represent reality.

Part 5: Apply the model to create spatial prediction layer

A. Create a subset of the predictor layer stack

1. Use the code below to create a subset of the predictor stack using the crop function
   i. Add and run the code below

```
# create subset area bounding box to limit raster processing time
subsetBox <- ext(390000,400000,4030000,4040000)

# Create a subset
predictorStackSubset <- crop(predictorStack, subsetBox)
```

B. Apply the model to create a continuous prediction raster

1. Use the predict function to apply the model, prediction stand total biomass using the predictor layers in the stack and the model.

```
#Predict the model for the subset area
predRaster <- predict(predictorStackSubset, rf2)
plot(predRaster)
```

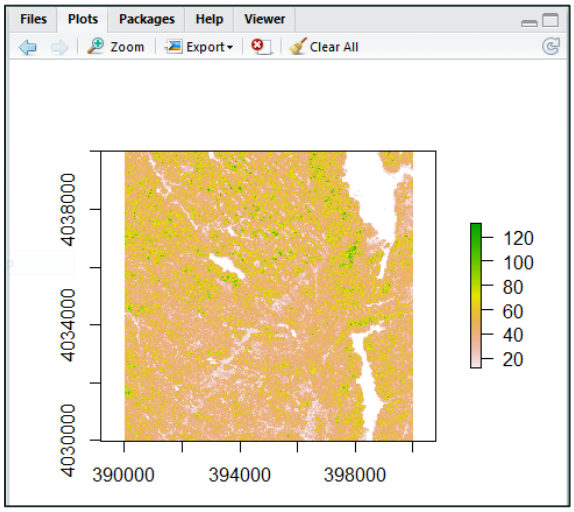

Figure 3. Total stand biomass as predicted by the second Random Forests regression model for a subset area on the North Kaibab.

In order to use the predict function, the layers in raster stack must have exactly the same names as the variable in the model dataset. If necessary, you can use the names() method to change the names of layers in a raster stack the same way you would change the names in a data frame.

C. Save the output prediction layer

1. Use the writeRaster() function to save the final prediction layer your course data folder
   i. If necessary, use the help function to pull up the documentation to get a reminder of the order of arguments, etc.

Congratulations! You have completed the final exercises and a complete spatial modeling workflow. You have explored multiple tools in R and practiced writing scripts to accomplish a variety of spatial data manipulation and analysis tasks.

Part 6: Appendix – Additional Resources

A. The following list of online resources provide more information on the topics covered in this exercise:

1. Spatial Data Analysis and Modeling with R: Spatial Distribution Models
2. Quantitative Methods in Spatial Ecology: Random Forest Species Distribution Modeling
3. Model Selection Approaches