University of the Philippines Manila
College of Arts and Sciences
Department of Physical Sciences and Mathematics

Ecological Niche Modelling of Dipterocarp Trees Using Maximum Entropy Methodl

A special problem in partial fulfillment
of the requirements for the degree of
Bachelor of Science in Computer Science

Submitted by:

Adrian Jose B. Sabado

June 2017

Permission is given for the following people to have access to this SP:

Available to the general public	Yes
Available only after consultation with author/SP adviser	No
Available only to those bound by confidentiality agreement	No

ACCEPTANCE SHEET

The Special Problem entitled "Ecological Niche Modelling of Dipterocarp Trees Using Maximum Entropy Methodl" prepared and submitted by Adrian Jose B. Sabado in partial fulfillment of the requirements for the degree of Bachelor of Science in Computer Science has been examined and is recommended for acceptance.

Richard Bryann L. Chua, M.Sc.
Adviser

EXAMINERS:

	Approved	**Disapproved**
1. Gregorio B. Baes, Ph.D. (*cand.*)	_____	_____
2. Avegail D. Carpio, M.Sc.	_____	_____
3. Perlita E. Gasmen, M.Sc. (*cand.*)	_____	_____
4. Marvin John C. Ignacio, M.Sc. (*cand.*)	_____	_____
5. Ma. Sheila A. Magboo, M.Sc.	_____	_____

Accepted and approved as partial fulfillment of the requirements for the degree of Bachelor of Science in Computer Science.

Ma. Sheila A. Magboo, M.Sc.
Unit Head
Mathematical and Computing Sciences Unit
Department of Physical Sciences and Mathematics

Marcelina B. Lirazan, Ph.D.
Chair
Department of Physical Sciences and Mathematics

Leonardo R. Estacio Jr., Ph.D.
Dean
College of Arts and Sciences

Abstract

A program that would predict the likely distribution of a species over a geographic area have always been a sought for tool in the field of ecology due to the wide variety of applications that it could have or as it is more commonly known, Niche Modelling. The species of interest, Dipterocarps, are medium to large, resinous forest trees belonging to tropical plant family dipterocarpaceae. Being one of the main source of timber in Asia, a suitability tool for dipterocarps would have some economic impact for the Philippines' export industry. For modelling its distribution, maximum entropy approach is used. Environmental variables used were obtained from PAG ASA but since localities are from central visayas mainly only, only the weather stations surrounding the localities are selected.The model produced has an AUC score 99.64% however, it is important we note that the presence data and climate data are based mainly from Central Visayas. For future research purposes, the software produced also includes a function to produce your own model using your own set of environmental variables. The software is using java as its interface and R (with Dismo as the main package) for its backend.

Keywords: R, Dismo, Maximum Entropy, Niche Modelling, Dipterocarpaceae

Contents

List of Figures

List of Tables

I. Introduction

A. Background of the Study

Dipterocarps are medium to large, resinous forest trees belonging to the tropical plant family Dipterocarpaceae. Dipterocarps comprise the main timber trees of tropical Asia and are ecologically major components of various lowland rain forests [1][2][3].

The Philippines is a net exporter of forest products, and forestry has long been an important part of its export economy. In 1968, forestry contributed 32.5 percent of total foreign exchange earnings. This rate dropped steadily, however, and in 1974 forestry accounted for only 10 percent of the total. A top dollar earner for many years, forestry products now rank behind nontraditional manufactures and mineral, coconut, and sugar products [4].

Discussions on the loss of forest in the Philippines always highlights the loss of the high-value dipterocarp forests which used to cover about 45 percent of the countrys total land area, exploited to provide timber for industries [5][6]. Today, the dipterocarps together with the other forest types molave, pine, mangrove, beach and mossy have become a scarce source of timber [5].

With the rapid decline on the number of our forests, especially the dipterocarps, the government made various attempts to address this issue. Data from 2003 to 2007 revealed that at least 500, 000 ha were restored forests as a result of past reforestation, afforestation, and regeneration of degraded natural forests [5][7].

In these efforts of reforestation, a tool that would predict the suitability of dipterocarp forest in the locations in the Philippines would help in determining which areas would be best suited for dipterocarp replanting.

This is where ecological niche modelling comes in. Ecological niche modelling is the process of using presence data and if possible, absence data, of a species, and a set of environmental variables to create a heat map of sorts that shows the likelihood of occurrence of a species in a certain geographic region.

Currently there's a lot of existing modelling methods but for this paper we're going to talk about the Maximum Entropy approach or as it is more commonly known, MaxEnt.

The best approoach in approximating an unknown probability distribution is to ensure that the approximation satisfies any constraints on the unknown distribution that we are aware of, and that subject to those constraints, the distribution should have maximum entropy [8].

B. Statement of the Problem

Making a tool that would display habitat suitability of dipterocarp trees in the Philippines through a visualization of the current species distribution would help the governments efforts of reforestation.

This tool would be essential in planning reforestation activities so as not to waste resources on failed attempts.

Last 2015, the DENR funded a 7.2 billion National Greening Program (NGP) and land survey project as stated by COA. The tree planting project was said to be unsuccessful. They attempted to pursue the project without having an efficient system to conduct the implementation and monitoring. According to COA, the DENR identified non-plantable areas as tree planting sites because it did not conduct mapping and planning [9].

In order to avoid these kinds of incidents from happening again, planning would be crucial and in this, a tool that predicts habitat suitability would help immensely.

C. Objectives of the Study

This study would produce a software that would take in presence data of a dipterocarp species and environmental variables (climatic variables, soil information, etc) from all over the Philippines. Presence data is inputted through a csv file containing the latitude and longitude where a certain species is located. Using the

presence data and environmental variables, the tool would then output a species distribution showing the habitat suitability of the species selected in the Philippines, where it is likely and unlikely to thrive. The output would be a visual model of the distribution. The functional specs of the software are as follows:

1. Allows the user to
   (a) Input the presence data of the selected species
   (b) Input the environmental variables (also called layers) of the geographic region (the Philippines)
   (c) Filter the heatmaps produced, by region so only the region selected is displayed.
2. The program shall be able to
   (a) Use the entered presence data and environmental variables to predict the species distribution
   (b) Be able to output the percentage contribution of each environmental variable to the model created
   (c) Be able to output a series of distributions in a GIF like manner which would be the series of distributions from current time up to the entered X years from now
   (d) Display the average likelihood of the species in a region over the entered X years.

D. Significance of the Project

The efforts for reforestation of the Philippine government along with the private sectors is slowly but surely taking effect. From the 5.40 million ha forest coverage to 7.20 million ha forest coverage in 2003, it can be seen that it is improving [5].

The software can be used to help in further improving forest coverage of dipterocarp tress in the Philippines. This tool would help in the planning stage of a

reforestation activity by showing the suitable places where the dipterocarp species of interest is likely to flourish.

Then, when the forest coverage is sustainable, this tool may then be used to determine locations where a dipterocarp tree species is likely to be found and use this information to improve timber harvesting and timber export industry thereby contributing to the overall improvement of the Philippine economy.

E. Scope and Limitations

The software produced in this study will have the following constraints:

1. It won't be able to process presence-absence or presence/pseudo-absences data or other types of localities as input
2. It will project the model onto the Philippines only
3. It will be able to process presence data from the Philippines only
4. Environmental variables must be of the same space granularity (same area per cell) and extent. Extent is the covered area of the environmental variable defined by maximum and minimum latitude and longitude coordinates.
5. Environmental variables used in the built in model consists of climate data from PAG ASA Specifically, they are the minimum temperature, maximum temperature, mean temperature, average amt of rainfall, average station pressure, and relative humidity.
6. Presence data used in the built in model consists of localities provided by DENR
7. The accuracy of data, both environmental variables and presence data, is outside of the system

F. Assumptions

The software produced by this study works under the following assumptions:

1. The user has the dataset ready, this study doesnt include the dataset gathering

2. The user is assumed to be knowledgeable in niche modelling and thus, knows how to interpret the output of the software

II. Review of Related Literature

The ecological niche of a species can be defined as those ecological conditions under which it can maintain populations without immigration [10]. Ecological niches and associated potential geographic ranges can be approximated using correlative algorithms that by relating known point-occurrence data to digital GIS data layers, summarize spatial variations in these layers in multidimensional environmental space [11].

Predicting these ecological niches through a correlative approach is the aim of ecological niche modelling. Through presence data and various environmental variables as layers, one is able to statistically determine the likelihoods of a certain species being found in a certain geographic region.

Through time, various methods for ecological niche modelling appeared. Two of the most popular are MaxEnt (maximum entropy method) and GARP (Genetic Algorithm for Rule-Set Prediction).

In 2006, a study was made comparing the different modelling methods available to test the strengths of these methods under different assumptions. In this study, it was shown that Maxent methods ranked 3rd when the mean AUC (area under curve, a test of accuracy of modelling methods) over all species are considered [12]. Among the presence only modelling methods, Maxent ranked the highest.

In 2007, another study was made comparing the predictive success of the two. In the sampled region test, Maxent got an AUC (area under the curve) score of 0.733 compared to the 0.608 of GARP, suggesting that GARP are less predictive models than Maxent [13].

According to Khosravi et al. [14], data on species absence are often unavailable or unreliable for threatened species. Thus, modelling techniques that require presence-only data such as Maxent have been widely used to predict habitat distributions. It only requires species occurrence points, it uses continuous and categorical data and the interactions between them, it is not sensitive to collinearity between between environmental variables, the resulting probability distributions

are easy to analyze, overfitting can be avoided by using regularization, and it is very robust at detailed scales [14].

A paper on 2006 [15], showed how Maxent works and how it avoids overfitting through regularization. It is also shown that Maxent doesn't put the same weight on the environmental variables (essentially saying that each environmental variable contributes different weights to the predictive accuracy of the model) [15], which is true to the real world applications.

In 2007, Pearson et al. showed that even with a small number of localities or presence data, current modelling techniques Maxent and GARP can still produce statistically significant models, however, they noted by the end of paper, that such models must be interpreted with care [16].

Then another paper in 2010, showed that MARS (multivariate adaptive regression splines), another modelling technique for species distribution, had a decrease on model reliability with the decrease in sample size [17]. This further supports the use of presence only methods like Maxent. Later in the paper, it is concluded that the model's predictive power improves greatly only at 18-20 unique presences compared to the greater than 5 presences for Maxent in [16].

With the wide variety of modelling methods available as of now, different efforts for restoration, conservation, cultivation, etc. would benefit greatly.

In particular, the Philippines has been trying to restore its forest coverage after its rapid decline reaching a low of 5.40 million ha of forest coverage in the late 1980's from the 17.00 million ha of forest coverage during the early 1930s [5].

But due to the government's efforts along with the private sectors and NGOs, it increased and reached 7.20 million ha in 2003. The number of hotspots for illegal logging activities in natural forests was reduced by 78 percent [5].

However, in 2015, a P7 billion tree-planting project of DENR resulted in failure. And according to COA (Commission on Audit), the DENR identified "non-plantable areas" as tree-planting sites because it did not conduct mapping and planning [9].

In projects like this, a tool, specifically a niche modelling tool, would help in determining habitat suitability of a species making the planning and decisions of project planners more informed.

III. Theoretical Framework

A. Ecological Niche Modelling

Prediction of species distributions is central to diverse applications in ecology, evolution and conservation science. There is increasing electronic access to vast sets of occurrence records in museums and herbaria, yet little effective guidance on how best to use this information in the context of numerous approaches for modelling distributions [12].

Ecological Niche Modelling (ENM) is the process of using the said information, specifically localities or presence data and a set of environmental variables (also called layers) to produce a prediction model that can be used to predict habitat suitability of input species over a geographical area.

ENM works under the assumption that:

1. The current species distribution (represented by the localities of the species) is a good approximator of future distributions

2. The inputted data came from a reliable source

Models produced from this process are used to predict species' invasion, plan where to put reservation areas, discover potential locations of niches of a specific species.

B. Dipterocarp

Dipterocarps are medium to large, resinous trees belonging to the tropical plant family Dipterocarpaceae [18]. It comprise the main timber trees of tropical Asia and are ecologically major components of various types of lowland rain forests [1].

Many dipterocarps are big trees that have stem diameters of up to 200 cm or more and often reach the top of forest canopy.

Sometimes, dipterocarps can form extensive patches in the forest with some even reaching 89.4% coverage on the lower slopes of Mt Silay on Negros Island

[18].

C. Maximum Entropy Approach

MaxEnt is a modelling approach that uses the principle of maximum entropy inspired from thermodynamics. It takes a list of species presence locations as input, often called presence-only (PO) data, as well as a set of environmental predictors (e.g. precipitation, temperature) across a user-defined landscape that is divided into grid cells [19].

The principle of maximum entropy (MaxEnt) provides a means to obtain least-biased statistical inference when insuficient information is available. Its idea is to estimate a target probability distribution by finding the probability distribution of maximum entropy subject to a set of constraints that represent our incomplete information about the target distribution [20].

As J.W. Gibbs (1878) [21] said before, a system (in our case, a niche of a species) that has achieved its maximum entropy will be in a state of equilibrium. That is, after some time (entropy increases with time - Second Law of Thermodynamics), the system will follow the distribution which has the maximum entropy.

The approach consists of formalizing the constraints on the unknown probability distribution π in the following way. We assume that we have a set of known real-valued functions f_1,...,f_n on X, known as "features" (which for our application will be environmental variables or functions thereof). We assume further that the information we know about π is characterized by the expectations (averages) of the features under π [15].

Here, each feature f_j assigns a real value $f_j(x)$ to each point x in X. The expectation of the feature f_j under π is defined as

$$\sum_{(x \epsilon X)} \pi(x) f_j(x)$$

and denoted by $\pi[f_j]$. In general, for any probability distribution p and function f, we use the notation p[f] to denote the expectation of f under p.

The feature expectations $\pi[f_j]$ can be approximated using a set of sample points x_1,..., x_m drawn independently from X (with replacement) according to the probability distribution π.

The empirical average of f_j is

$$1/m \sum_{(i=1)}^{m} = f_j(i)$$

which we can write as $\hat{\pi}[f_j]$ (where $\tilde{\pi}$ is the uniform distribution on the sample points), and use as an estimate of $\pi[f_j]$.

By the maximum entropy principle, therefore, we seek the probability distribution $\hat{\pi}$ of maximum entropy subject to the constraint that each feature f_j has the same under $\hat{\pi}$ as observed empirically, i.e.

$$\hat{\pi}[f_j] = \tilde{\pi}[f_j], for each feature f_j$$

Consider all probability distributions of the form

$$q_{\boldsymbol{\lambda}}(x) = \frac{e^{\boldsymbol{\lambda} \cdot \boldsymbol{f}(x)}}{Z_{\boldsymbol{\lambda}}}$$

where λ is a vector of n real-valued coefficients or feature weights, f denotes the vector of all n features, and Z_λ is a normalizing constant that ensures that q_λ sums to 1. Such distributions are known as Gibbs distributions. Convex duality shows that the Maxent probability distribution $\hat{\pi}$ is exactly equal to the Gibbs probability distribution q_λ that maximizes the likelihood of the m sample points. Equivalently, it minimizes the negative log likelihood of the sample points

$$\tilde{\boldsymbol{\pi}}[-\ln(q_{\boldsymbol{\lambda}})]$$

which can also be written

$$\ln Z_\lambda - \frac{1}{m} \sum_{(i=1)}^{m} = \lambda(f(x_i))$$

and termed the "log loss"

However, it can be seen that since Maxent constraints the model to fit the observed moments, issues of overfitting will surely arise. This is where regularization comes. This can be done by relaxing the constraint of exacting the moments of the observed localities, replacing the constraint to

$$|\hat{\pi}[f_j] - \tilde{\pi}[f_j]| \leq \beta_j, \quad \text{for each feature } f_j$$

for some constants β_j. This would make the Maxent distribution be the Gibbs distribution that minimizes

$$\tilde{\pi}[-\ln(q_\lambda)] + \sum_j \beta_j |\lambda_j|$$

where the first term is the log loss while the second term penalizes the use of large values for the weights λ_j. Regularization forces Maxent to focus on the most important features, and l_1-regularization tends to produce models with few non-zero λ_j values [22]. Such models are less likely to overfit, because they have fewer parameters; as a general rule, the simplest explanation of a phenomenon is usually best [15].

IV. Design and Implementation

A. Use Cases

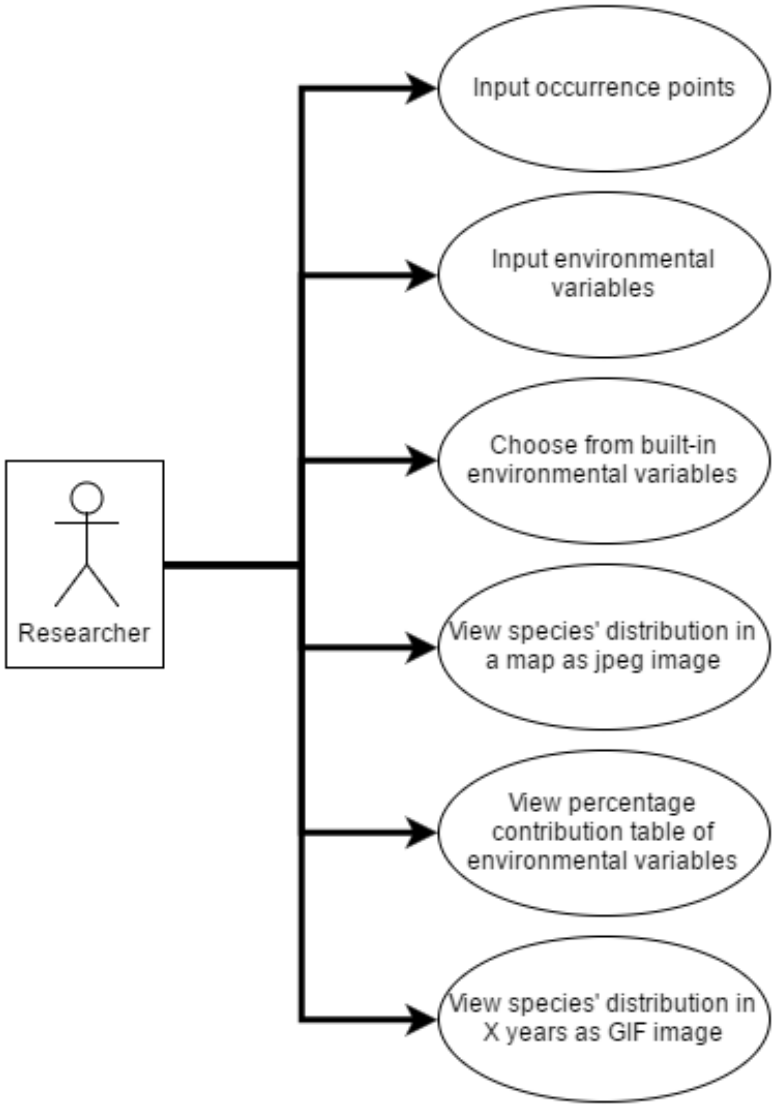

Figure 1: Use Case Diagram

The researcher will input occurrence data (csv format) and environmental variables (asc files). The tool will use these to produce a prediction model through the Maxent method.

After the computation for the prediction model, the tool will now then use this model to create a map visualization of the model, using the difference in colors as hint of increasing likelihood of occurrence. The outputted map will be in the

form of a jpeg image. The map used as reference would be the Philippine map.

B. Flowchart Diagram - Researcher

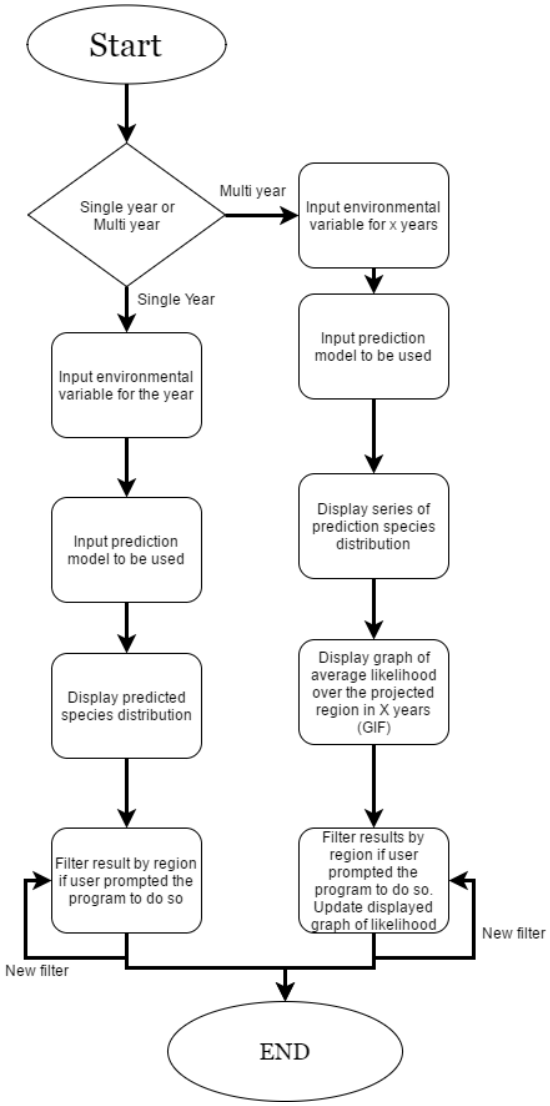

Figure 2: Flowchart Diagram - Researcher

The first step in using the tool (if you are a researcher) is to input both the model to be used and environmental data. It contains the species' distribution over the Philippine map, showing where a species is likely to be found through a color scale.

On the final phase of processing, the tool would also generate the percentage contribution for each of the environmental variables used as input. This would show how much each environmental variable contributed to the model, also show-

ing how relevant an environmental variable is to the distribution of the species generated.

C. Flowchart Diagram - AI Expert

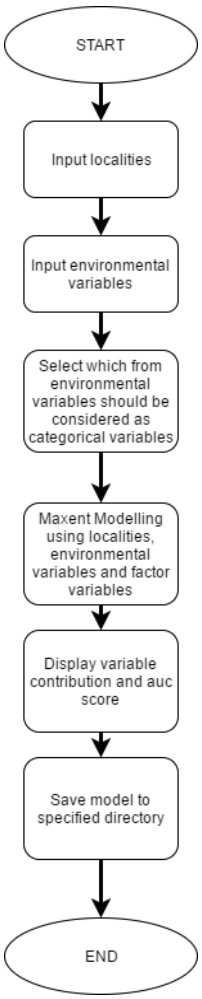

Figure 3: Flowchart Diagram - AI Expert

For researchers that would be in charge of remodelling, the flow would be first inputting the localities of the dipterocarp species and the environmental variables. An option to select which environmental variables are to be treated as categorical variables will also be available to the modeller. The program would also ask for the directory where the model would be saved. Modelling starts when all required inputs are given and the user press the start modelling button. When the modelling is finished, the contribution per variable would be displayed to the

user then the model would be saved to the specified directory.

V. Results

A. General User Functionalities

The first tab that the user would see is the single year prediction tab. Aside from the single year prediction tab, the user are also given two more tabs of differing functionalities, the multi year prediction and remodelling.

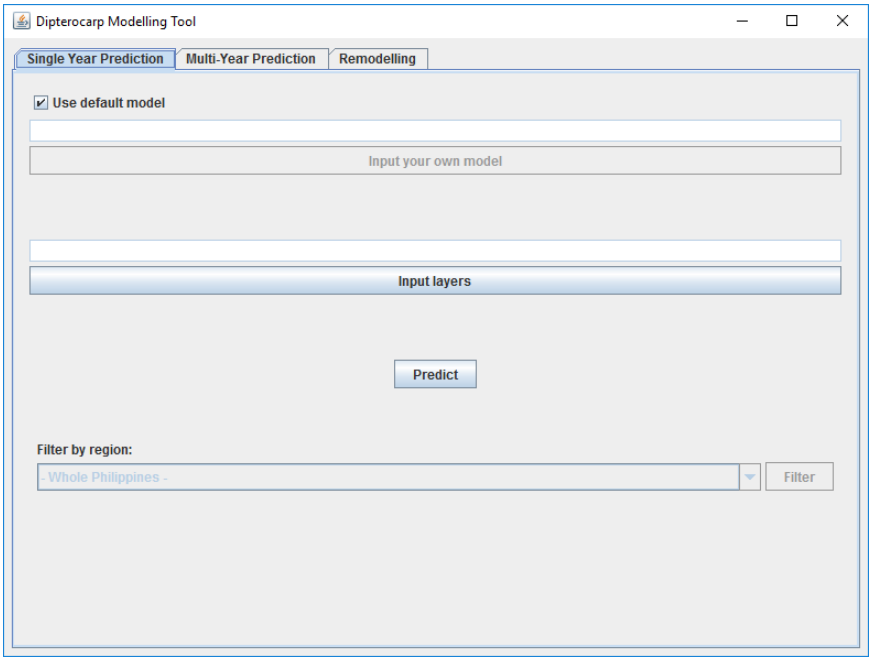

Figure 4: Single Year Prediction

A..1 Single Year Prediction Using The System's Model

If the user is a researcher, the figure below shows how the researcher could use the system's built-in model to predict the distribution of species along with his/her environmental variables. The filter functionality is initially disabled as this feature would be for filtering the results (which isn't produced yet) onto a specific region. Clicking the predict button would initiate the prediction process.

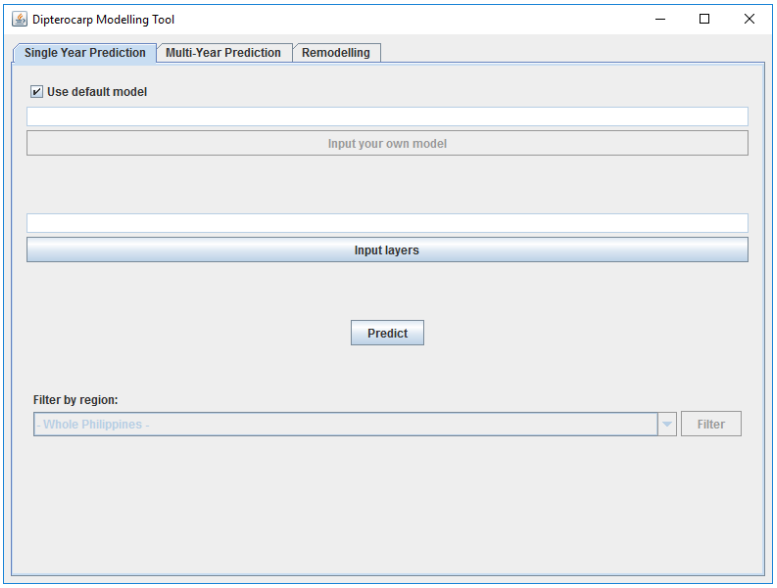

Figure 5: Single Year Prediction - Built in Mod

A..2 Single Year Prediction Using A User Uploaded Model

Another option available to the user is to predict using his/her own model pre-trained (using R's Dismo package) along with the same environmental variables. As is before, the filter function is disabled yet for the same reason.

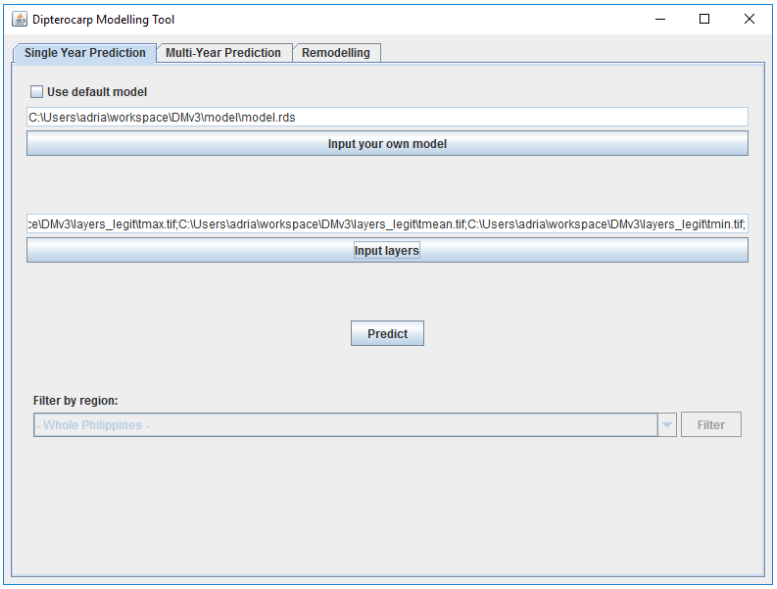

Figure 6: Single Year Prediction - User Uploaded Model

A..3 Single Year Prediction Results

After the program has finished the prediction process, the filter functionality would be enabled. The filter function, filters the latest prediction done using the selected region. After that, a popup window would appear showing the predicted distribution of the species over the selected region (default is whole Philippines). Displayed result supports zooming and panning via mouse actions.

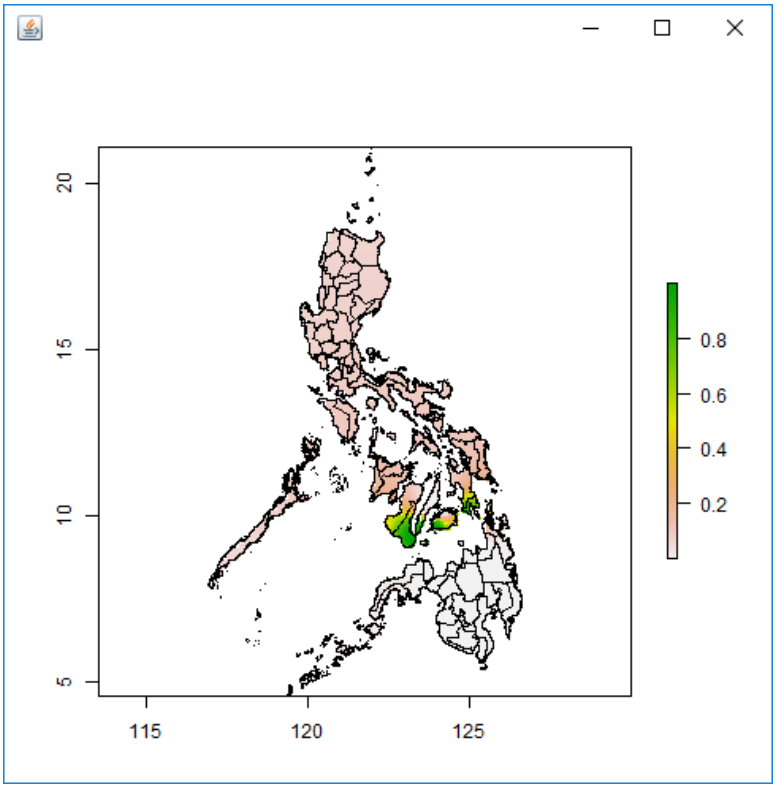

Figure 7: Single Year Prediction - Results

A..4 Single Year Prediction Filter

The filter function filters the latest successful prediction using the currently selected region. A new popup window displaying the filtered result would appear.

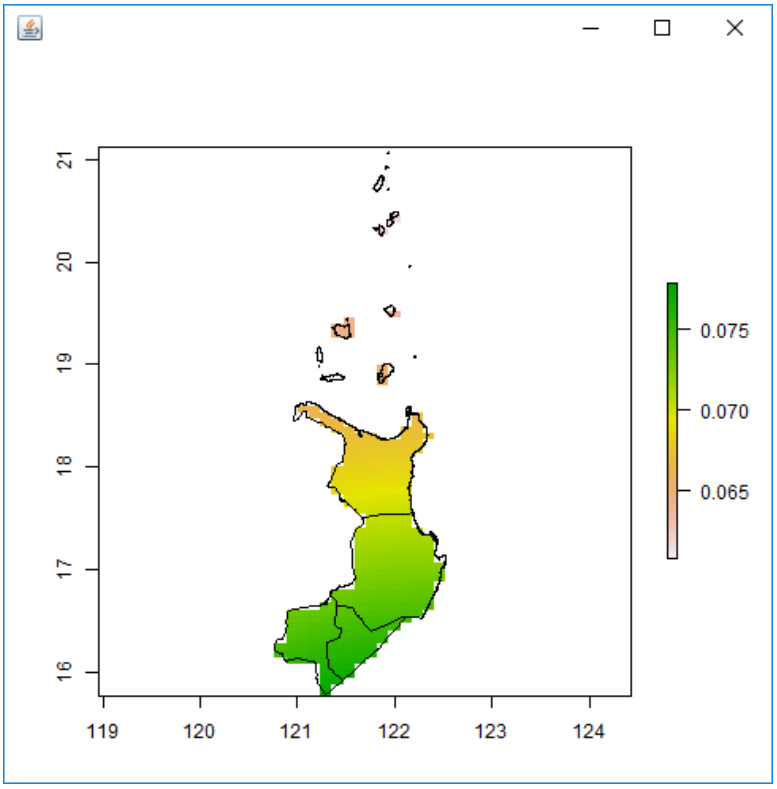

Figure 8: Single Year Prediction Filter

A..5 Multi Year Prediction Using The System's Model

Like the single year using the system's model, the user also has the option to use the system's model for multi year predicting. However, this time around a different set of environmental variables are needed to be inputted for every year.

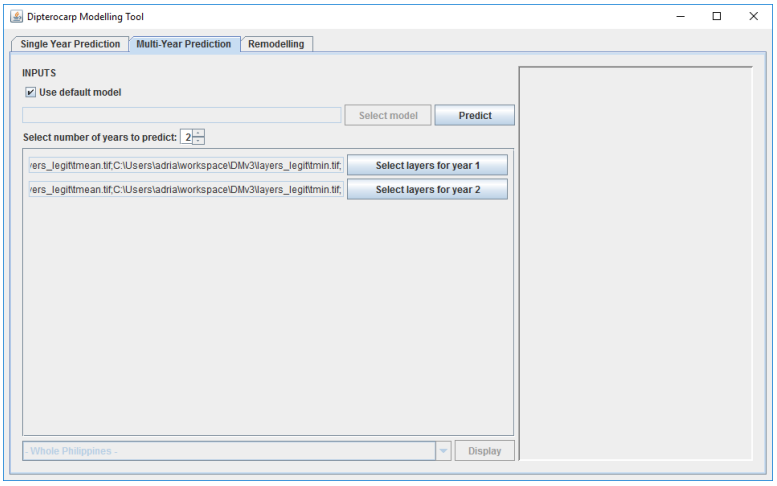

Figure 9: Multi Year Prediction - Built in Model

A..6 Multi Year Prediction Using A User Uploaded Model

As with single year prediction, you can also select your own model for the multi year prediction.

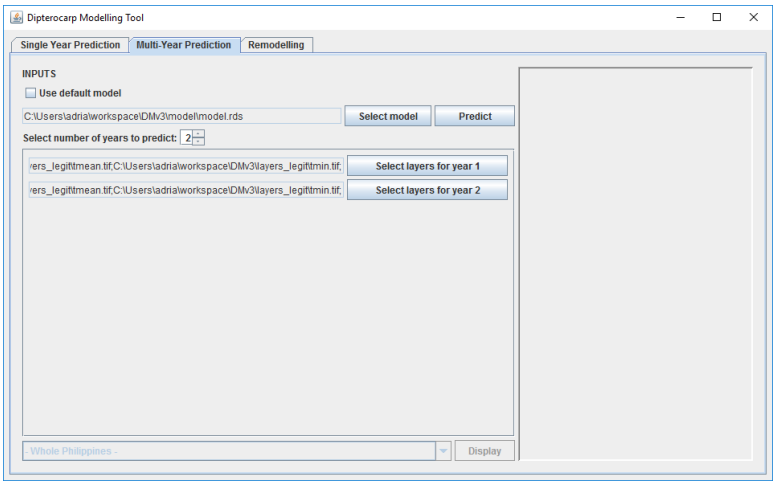

Figure 10: Multi Year Prediction - User Uploaded Model

A..7 Multi Year Prediction Results

Added results on the multi year prediction would be the graph of likelihood over time over the currently projected region. This result also changes as you use a different filter. The program would display in a pop up a GIF like series of images containing the distribution of species for the different years.

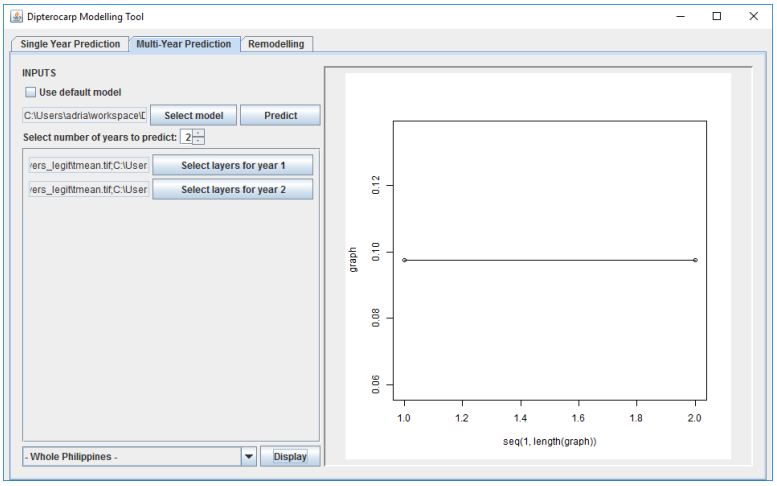

Figure 11: Multi Year Prediction - Graph

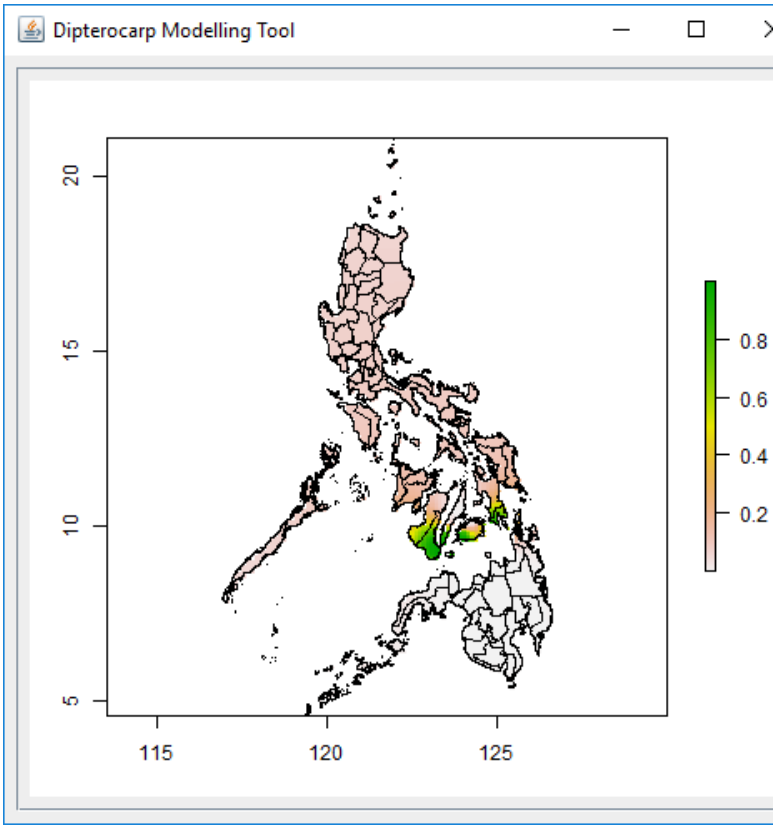

Figure 12: Multi Year Prediction - Results

A..8 Multi Year Prediction Filter

Filtering in the multi year prediction tab would update not only the series of pictures displayed, but also the graph of likelihood displayed.

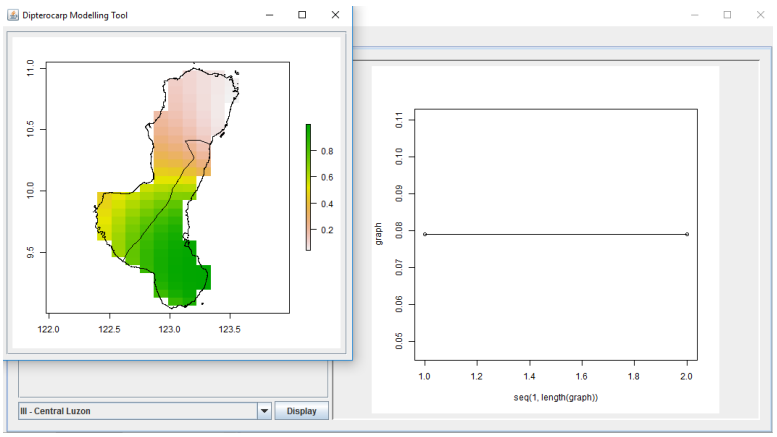

Figure 13: Multi Year Prediction - Filter

A..9 Remodelling

When the researcher's task is to remodel, the inputs asked by the program would be the presence data, the layers, the destination, and if it exists the layers that is to be considered as categorical layers. Clicking the start modelling button would initiate the modelling process.

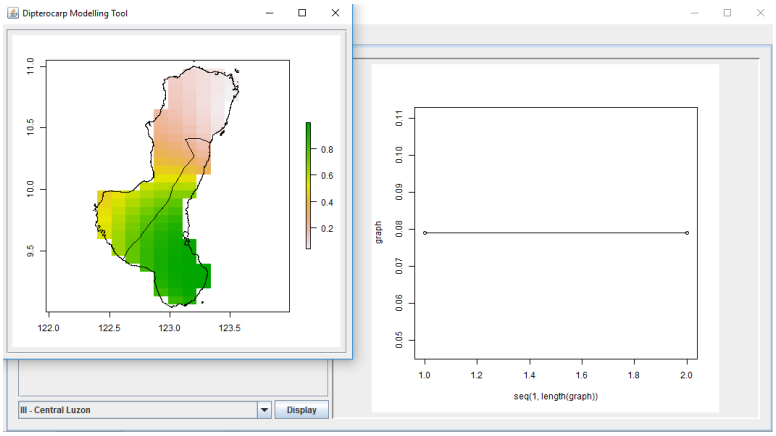

Figure 14: Multi Year Prediction - Filter

A..10 Remodelling - Result

When the modelling is finished, the modelling object is then saved to the specified directory with the specified filename. Should the user wish to view the variable contributions of the produced model, the user may click the show variable contribution button.

VI. Discussions

The reason for the choice of environmental variables used in the system's built in model is due to the available data as of the time of study. This is why all of the environmental variables used are meteorological data from PAG ASA. The proponent have found no available data on edaphic (soil information) variables, which could be even more significant than the climate data. Looking at the variable contributions, we can see that despite having six variables, only 2 have contributions with one of them being contributing significantly more to the model than the other one.

Another problem on the dataset is that the only available source of data contains localities on Central Visayas only even though the program is capable of handling data for the whole Philippines (the program would run relatively slower though) indicating that the program/tool is not being used to its full potential.

From the AUC score in the table below, we can see that the model have a score of 99.64%. However, there are some problems with this. As stated before the localities are only from Central Visayas, what more is that the localities are relatively close to one another which could have an effect on the model by making the surrounding area overrepresented, having an effect on the weights that would be produced.

Description	Result
AUC Score	99.64%
Tmin Contribution	88.2043%
Tmax Contribution	11.7957%

VII. Conclusions

The software produced in this study is a stand-alone application that aims to provide researchers with a way to create models that could be used to predict the habitat suitability of Dipterocarp trees. Aside from it, the software also gives the user an existing model that could already be used to predict habitat suitability through data provided by PAG ASA and DENR.

The software uses a machine learning algorithm, MaxEnt, to predict the likely species distribution of Dipterocarp trees. It is able to achieve 99.64% accuracy in terms of AUC in predicting the suitable habitat for Visayas. Do note however, that the data used for the built in model are only from Central Visayas, so predictions using the built in model is accuracte only within the vicinity of the region.

VIII. Recommendations

The proponent suggests that the built in model be improved by collecting more presence data of Dipterocarp species nationwide as well as collecting more climate data and if possible soil information (altitude and soil type). As according to [18], these variables could characterize more the presence of Dipterocarp species.

A way to view what model could be used given a set of environmental variables would also be handy as from the current constraints, you could only predict using the same type of variables you used training your model. Not having to guess which environmental variables are used for which models would save a lot of time and avoid confusion.

For the implementation aspect, the proponent suggests moving the software online. An online implementation of the said software would be more accessible to more researchers, benefitting more people. Aside from being accessible to more people, it would also be available over a larger area as you don't need to have the software installed on your machine.

IX. Bibliography

[1] P. S. Ashton *et al.*, *Dipterocarpaceae.*, vol. 9. 1982.

[2] T. Whitmore, "Tropical rain forests of the far east, 2nd edn. clarendon," 1984.

[3] M. Newman, P. Burgess, and T. Whitmore, "Manuals of dipterocarps for foresters. borneo island light hardwoods," *Center for International Forestry Research, Jakarta*, 1996.

[4] E. L. Boado, "Incentive policies and forest use in the philippines," *Public policies and the misuse of forest resources*, pp. 165–204, 1988.

[5] E. Guiang and R. Aragon, "Forest restoration at the landscape level in the philippines," *Forest landscape restoration for Asia-Pacific forests*, p. 125, 2016.

[6] A. Revilla Jr., "The worlds dipterocarp forests. in forest renewal in actual practice," *Columbian Publishing Corporation*, 1984.

[7] Ecosystem Research and Development, "Development and management of forest plantations: a guidebook.," 2010.

[8] E. T. Jaynes, "Information theory and statistical mechanics," *Physical review*, vol. 106, no. 4, p. 620, 1957.

[9] M. Ramos, "COA: P7b tree-planting project of denr a failure." http://newsinfo.inquirer.net/687426/coa-p7b-tree-planting-project-of-denr-a-failure. Retrieved: 2015-05-24.

[10] J. Grinnell, "The niche-relationships of the california thrasher," *The Auk*, vol. 34, no. 4, pp. 427–433, 1917.

[11] A. Guisan and N. E. Zimmermann, "Predictive habitat distribution models in ecology," *Ecological modelling*, vol. 135, no. 2, pp. 147–186, 2000.

[12] R. P Anderson, M. Dudík, S. Ferrier, A. Guisan, R. J Hijmans, F. Huettmann, J. R Leathwick, A. Lehmann, J. Li, L. G Lohmann, *et al.*, "Novel methods improve prediction of species distributions from occurrence data," *Ecography*, vol. 29, no. 2, pp. 129–151, 2006.

[13] A. Townsend Peterson, M. Papeş, and M. Eaton, "Transferability and model evaluation in ecological niche modeling: a comparison of garp and maxent," *Ecography*, vol. 30, no. 4, pp. 550–560, 2007.

[14] R. KHOSRAVI, M.-R. HEMAMI, M. MALEKIAN, A. L. FLINT, and L. E. FLINT, "Maxent modeling for predicting potential distribution of goitered gazelle in central iran: the effect of extent and grain size on performance of the model," *Turkish Journal of Zoology*, p. 40, 2016.

[15] S. J. Phillips, R. P. Anderson, and R. E. Schapire, "Maximum entropy modeling of species geographic distributions," *Ecological modelling*, vol. 190, no. 3, pp. 231–259, 2006.

[16] R. G. Pearson, C. J. Raxworthy, M. Nakamura, and A. Townsend Peterson, "Predicting species distributions from small numbers of occurrence records: a test case using cryptic geckos in madagascar," *Journal of biogeography*, vol. 34, no. 1, pp. 102–117, 2007.

[17] R. G. Mateo, Á. M. Felicísimo, and J. Muñoz, "Effects of the number of presences on reliability and stability of mars species distribution models: the importance of regional niche variation and ecological heterogeneity," *Journal of Vegetation Science*, vol. 21, no. 5, pp. 908–922, 2010.

[18] E. S. Fernando, M. J. Bande, R. A. Piollo, D. D. Sopot, N. E. Dolotina, W. G. Granert, P. P. Milan, M. J. C. Ceniza, and M. T. Pollisco, "Habitats of philippine dipterocarps," 2009.

[19] C. Merow, M. J. Smith, and J. A. Silander, "A practical guide to maxent for modeling species distributions: what it does, and why inputs and settings matter," *Ecography*, vol. 36, no. 10, pp. 1058–1069, 2013.

[20] K. Garcia, R. Lasco, A. Ines, B. Lyon, and F. Pulhin, "Predicting geographic distribution and habitat suitability due to climate change of selected threatened forest tree species in the philippines," *Applied Geography*, vol. 44, pp. 12–22, 2013.

[21] J. W. Gibbs, "On the equilibrium of heterogeneous substances," *American Journal of Science*, no. 96, pp. 441–458, 1878.

[22] P. M. Williams, "Bayesian regularization and pruning using a laplace prior," *Neural computation*, vol. 7, no. 1, pp. 117–143, 1995.

X. Appendix

A. Source Code

```
#include <iostream>

using namespace std;

int main{
        cout << "Hello world!" << endl;
        return 0;
}
```

A..1 UI

```
package ui;

import java.awt.Color;
import java.awt.Dimension;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.image.BufferedImage;
import java.io.File;
import java.io.IOException;
import java.util.ArrayList;

import javax.imageio.ImageIO;
import javax.swing.DefaultComboBoxModel;
import javax.swing.ImageIcon;
import javax.swing.JButton;
import javax.swing.JCheckBox;
import javax.swing.JComboBox;
import javax.swing.JFileChooser;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JPanel;
import javax.swing.JScrollPane;
import javax.swing.JSpinner;
import javax.swing.JTabbedPane;
import javax.swing.JTextField;
import javax.swing.SpinnerNumberModel;
import javax.swing.border.BevelBorder;
import javax.swing.event.ChangeEvent;
import javax.swing.event.ChangeListener;

import org.rosuda.REngine.Rserve.RConnection;
import org.rosuda.REngine.Rserve.RserveException;

import componentActions.Enable_Own_Model;
import componentActions.MP_Predict;
import componentActions.Remodel_Prep;
import componentActions.SP_Predict;
import componentActions.Select_EV;
import componentActions.Select_Model;
import javaBackend.Filter_Results;
import javaBackend.ImageDisplayer;
import javaBackend.MP_Displayer;
import javaBackend.Region;
import javaBackend.RegionParser;
import net.miginfocom.swing.MigLayout;
import rBackend.RConnector;

public class Main {

        private JFrame frame;
        private JPanel spPanel;
        private JPanel spInputs;
        private JCheckBox spDefModCB;
        private JTextField spModTF;
        private JButton spModBtn;
        private JTextField spEVTF;
        private JButton spEVBtn;
        private JButton spPredict;

        private RConnection c;
        private JPanel mpPanel;
        private JPanel mpInputs;
        private JLabel lblInputs;
        private JCheckBox mpDefModCB;
        private JTextField mpModTF;
        private JButton mpModBtn;
        private JLabel lblSelectNumberOf;
        private JSpinner spinner;
        private JPanel mpEVPanel;

        private ArrayList<Region> regions;

        private JTextField mpYear1EVTF;
        private JTextField mpYear2EVTF;
```

```
private ArrayList<JTextField> mpEVTF = new ArrayList<JTextField>();
private ArrayList<JButton> mpEVBtn = new ArrayList<JButton>();

private int envDataAdded = 2;
private JScrollPane scrollPane;
private JPanel panel;
private JComboBox filterBox;
private JButton filterBtn;
private JLabel lblFilterByRegion;
private JTabbedPane displayPane;
private JButton mpPredict;
private JComboBox mpFilterBox;
private JButton btnDisplay;
private JPanel graphPanel;
private JPanel modPanel;
private JTextField localitiesTF;
private JButton btnLocalities;
private JTextField layersTF;
private JButton btnLayers;
private JTextField destTF;
private JButton btnSelectDestination;
private JTextField factorsTF;
private JLabel factorsLbl;
private JButton btnStartModelling;
private JLabel lblModelParameters;
private JButton btnShowVariableContribution;


/**
 * Launch the application.
 */
public static void main(String[] args) {
        Main window = new Main();
        window.frame.setVisible(true);

}

/**
 * Create the application.
 */
public Main() {
        RConnector newConnection = new RConnector();
        RegionParser rp = new RegionParser();
        regions = rp.regionParser();
        Runtime.getRuntime().addShutdownHook(new Thread(new Runnable() {
            public void run() {
                try {
                                c.shutdown();
                                System.out.println("Rserve terminated");
                        } catch (RserveException e) {
                                // TODO Auto-generated catch block
                                JOptionPane.showMessageDialog(null, "Rserve didn't
                                     shutdown properly. Shut it down manually to
                                     deallocate resources", "Notice", JOptionPane.
                                     WARNING_MESSAGE);
                        }
            }
        }));
        this.c = newConnection.getRConnection();
        initialize();

}

/**
 * Initialize the contents of the frame.
 */
private void initialize() {
        frame = new JFrame();
        frame.setTitle("Dipterocarp Modelling Tool");
        frame.setMinimumSize(new Dimension(800, 600));
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.getContentPane().setLayout(new MigLayout("", "[grow]", "[grow]"));


        displayPane = new JTabbedPane(JTabbedPane.TOP);
        frame.getContentPane().add(displayPane, "cell 0 0,grow");

                spPanel = new JPanel();
                displayPane.addTab("Single Year Prediction", null, spPanel, null);
                spPanel.setLayout(new MigLayout("", "[grow]", "[grow]"));

                        spInputs = new JPanel();
                        spPanel.add(spInputs, "cell 0 0,grow");
                        spInputs.setLayout(new MigLayout("", "[grow][]",
                             "[][][][10%][][][10%][][grow]"));

                                spModTF = new JTextField();
                                spModTF.setBackground(Color.WHITE);
                                spModTF.setEditable(false);
                                spInputs.add(spModTF, "flowx,cell 0 1 2 1,growx");
                                spModTF.setColumns(10);

                                spModBtn = new JButton("Input your own model");
                                spInputs.add(spModBtn, "cell 0 2 2 1,grow");
                                spModBtn.setEnabled(false);
                                spModBtn.addActionListener(new Select_Model(spModTF));

                                spEVTF = new JTextField();
```

```
                spEVTF.setBackground(Color.WHITE);
                spEVTF.setEditable(false);
                spInputs.add(spEVTF, "cell 0 4 2 1,growx");
                spEVTF.setColumns(10);

                spEVBtn = new JButton("Input layers");
                spInputs.add(spEVBtn, "cell 0 5 2 1,grow");
                spEVBtn.addActionListener(new Select_EV(spEVTF));

                spDefModCB = new JCheckBox("Use default model");
                spInputs.add(spDefModCB, "cell 0 0");
                spDefModCB.setSelected(true);
                spDefModCB.addChangeListener(new Enable_Own_Model(
                    spDefModCB, spModBtn));

                panel = new JPanel();
                spInputs.add(panel, "cell 0 8 2 1,grow");
                panel.setLayout(new MigLayout("", "[grow][]",
                    "[15%][]"));

                lblFilterByRegion = new JLabel("Filter by region:");
                panel.add(lblFilterByRegion, "cell 0 1");

                filterBox = new JComboBox();
                filterBox.setEnabled(false);
                filterBox.setModel(new DefaultComboBoxModel(new String
                    [] {"- Whole Philippines -", "Negros Island Region
                    ", "National Capital Region", "Cordillera
                    Administrative Region", "I - Ilocos Region", "II -
                     Cagayan Valley", "III - Central Luzon", "IV-A -
                    CALABARZON", "MIMAROPA Region", "V - Bicol", "VI -
                     Western Visayas", "VII - Central Visayas", "VIII
                    - Eastern Visayas", "IX - Zamboanga Peninsula", "X
                     - Northern Mindanao", "XI - Davao Region", "XII -
                     Soccsksargen", "XIII - Caraga", "ARMM"}));
                panel.add(filterBox, "cell 0 2,growx");

                filterBtn = new JButton("Filter");
                filterBtn.setEnabled(false);
                panel.add(filterBtn, "cell 1 2");
                filterBtn.addActionListener(new SP_Filter_Results());

                spPredict = new JButton("Predict");
                spInputs.add(spPredict, "cell 0 7 2 1,alignx center");
                spPredict.addActionListener(new SP_Predict(spModTF,
                    spEVTF, spDefModCB, c, filterBtn, filterBox));

        mpPanel = new JPanel();
        displayPane.addTab("Multi-Year Prediction", null, mpPanel, null);
        mpPanel.setLayout(new MigLayout("", "[grow]", "[grow]"));

                mpInputs = new JPanel();
                mpPanel.add(mpInputs, "cell 0 0,grow");
                mpInputs.setLayout(new MigLayout("", "[grow][grow]",
                    "[][][][][][grow][]"));

                lblInputs = new JLabel("INPUTS");
                mpInputs.add(lblInputs, "cell 0 0");

                graphPanel = new JPanel();
                graphPanel.setBorder(new BevelBorder(BevelBorder.
                    LOWERED, null, null, null, null));
                mpInputs.add(graphPanel, "cell 1 0 1 6,grow");

                mpModTF = new JTextField();
                mpModTF.setEditable(false);
                mpInputs.add(mpModTF, "flowx,cell 0 2,growx");
                mpModTF.setColumns(10);

                mpModBtn = new JButton("Select model");
                mpModBtn.setEnabled(false);
                mpModBtn.addActionListener(new Select_Model(mpModTF));
                mpInputs.add(mpModBtn, "cell 0 2,wmin 100");

                mpDefModCB = new JCheckBox("Use default model");
                mpDefModCB.setSelected(true);
                mpDefModCB.addChangeListener(new Enable_Own_Model(
                    mpDefModCB, mpModBtn));
                mpInputs.add(mpDefModCB, "cell 0 1");

                lblSelectNumberOf = new JLabel("Select number of years
                     to predict:");
                mpInputs.add(lblSelectNumberOf, "flowx,cell 0 3");

                spinner = new JSpinner();
                spinner.setModel(new SpinnerNumberModel(new Integer(2)
                    , new Integer(2), null, new Integer(1)));
                mpInputs.add(spinner, "cell 0 3");

                scrollPane = new JScrollPane();
                mpInputs.add(scrollPane, "cell 0 4,grow");

                mpEVPanel = new JPanel();
                scrollPane.setViewportView(mpEVPanel);
                mpEVPanel.setLayout(new MigLayout("", "[grow]",
                    "[][]"));

                mpYear1EVTF = new JTextField();
                mpEVPanel.add(mpYear1EVTF, "flowx,cell 0 0,growx");
```

```
mpYear1EVTF.setColumns(10);
mpYear1EVTF.setEditable(false);

JButton mpYear1EVBtn = new JButton("Select layers for
    year 1");
mpEVPanel.add(mpYear1EVBtn, "cell 0 0,wmin 200");
mpYear1EVBtn.addActionListener(new Select_EV(
    mpYear1EVTF));

mpYear2EVTF = new JTextField();
mpEVPanel.add(mpYear2EVTF, "flowx,cell 0 1,growx");
mpYear2EVTF.setColumns(10);
mpYear2EVTF.setEditable(false);

JButton mpYear2EVBtn = new JButton("Select layers for
    year 2");
mpEVPanel.add(mpYear2EVBtn, "cell 0 1,wmin 200");
mpYear2EVBtn.addActionListener(new Select_EV(
    mpYear2EVTF));

mpEVTF.add(mpYear1EVTF);
mpEVTF.add(mpYear2EVTF);
mpEVBtn.add(mpYear2EVBtn);
mpEVBtn.add(mpYear1EVBtn);

mpFilterBox = new JComboBox();
mpFilterBox.setEnabled(false);
mpInputs.add(mpFilterBox, "flowx,cell 0 5,growx");
mpFilterBox.setModel(new DefaultComboBoxModel(new
    String[] {"- Whole Philippines -", "Negros Island
    Region", "National Capital Region", "Cordillera
    Administrative Region", "I - Ilocos Region", "II -
     Cagayan Valley", "III - Central Luzon", "IV-A -
    CALABARZON", "MIMAROPA Region", "V - Bicol", "VI -
     Western Visayas", "VII - Central Visayas", "VIII
    - Eastern Visayas", "IX - Zamboanga Peninsula", "X
     - Northern Mindanao", "XI - Davao Region", "XII -
     Soccsksargen", "XIII - Caraga", "ARMM"}));

btnDisplay = new JButton("Display");
btnDisplay.setEnabled(false);
mpInputs.add(btnDisplay, "cell 0 5,alignx right");
btnDisplay.addActionListener(new MP_Filter_Results());

mpPredict = new JButton("Predict");
mpInputs.add(mpPredict, "cell 0 2,wmin 100");
mpPredict.addActionListener(new MP_Predict(mpModTF,
    mpEVTF, mpDefModCB, c, mpFilterBox, btnDisplay,
    graphPanel));

modPanel = new JPanel();
displayPane.addTab("Remodelling", null, modPanel, null
    );
modPanel.setLayout(new MigLayout("", "[grow][]",
    "[10%][][][][][][50%,grow]"));

lblModelParameters = new JLabel("Model Parameters:");
modPanel.add(lblModelParameters, "cell 0 0");

localitiesTF = new JTextField();
localitiesTF.setEditable(false);
modPanel.add(localitiesTF, "flowx,cell 0 1,growx");
localitiesTF.setColumns(10);

btnLocalities = new JButton("Select localities");
btnLocalities.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
                JFileChooser chooser = new
                    JFileChooser();
                chooser.setDialogTitle("Select
                    localities");

                int userSelection = chooser.
                    showOpenDialog(frame);
                if(userSelection == JFileChooser.
                    APPROVE_OPTION) {
                    localitiesTF.setText(chooser.
                        getSelectedFile().
                        getAbsolutePath());
                }
        }
});
modPanel.add(btnLocalities, "cell 0 1,wmin 150");



layersTF = new JTextField();
layersTF.setEditable(false);
modPanel.add(layersTF, "flowx,cell 0 2,growx");
layersTF.setColumns(10);

btnLayers = new JButton("Select layers");
btnLayers.addActionListener(new Select_EV(layersTF));
modPanel.add(btnLayers, "cell 0 2,wmin 150");

btnSelectDestination = new JButton("Select destination
    ");
btnSelectDestination.addActionListener(new
    ActionListener() {
```

```
                                        public void actionPerformed(ActionEvent e) {
                                                JFileChooser chooser = new
                                                    JFileChooser();
                                                chooser.setDialogTitle("Select
                                                    destination of model object");

                                                int userSelection = chooser.
                                                    showSaveDialog(frame);
                                                if(userSelection == JFileChooser.
                                                    APPROVE_OPTION) {
                                                        File path = chooser.
                                                            getSelectedFile();

                                                        destTF.setText(path.
                                                            getAbsolutePath());
                                                }
                                        }
                                });

                                destTF = new JTextField();
                                destTF.setEditable(false);
                                modPanel.add(destTF, "flowx,cell 0 3,growx");
                                destTF.setColumns(10);
                                modPanel.add(btnSelectDestination, "cell 0 3,wmin
                                    150");

                                factorsLbl = new JLabel("Factors (hover mouse for
                                    tooltip):");
                                factorsLbl.setToolTipText("factor1, factor2, factor3")
                                    ;
                                modPanel.add(factorsLbl, "flowx,cell 0 4");

                                factorsTF = new JTextField();
                                modPanel.add(factorsTF, "cell 0 4,wmin 200");
                                factorsTF.setColumns(10);

                                btnStartModelling = new JButton("Start Modelling");
                                modPanel.add(btnStartModelling, "cell 1 1 1 3,wmin
                                    180,grow");

                                btnShowVariableContribution = new JButton("Show
                                    variable contribution");
                                btnShowVariableContribution.setEnabled(false);
                                btnShowVariableContribution.addActionListener(new
                                    ActionListener() {
                                        public void actionPerformed(ActionEvent arg0)
                                            {
                                                ImageDisplayer displayer = new
                                                    ImageDisplayer("pictures/weights.
                                                    png","model");
                                        }
                                });
                                modPanel.add(btnShowVariableContribution, "cell 1 4,
                                    wmin 180,grow");

                                btnStartModelling.addActionListener(new Remodel_Prep(c
                                    , layersTF, localitiesTF, factorsTF, destTF,
                                    btnShowVariableContribution));

                                SpinnerListener cl = new SpinnerListener();
                                spinner.addChangeListener(cl);
        }

        public class SpinnerListener implements ChangeListener {

                @Override
                public void stateChanged(ChangeEvent arg0) {
                        // TODO Auto-generated method stub
                        int newVal = (int) spinner.getValue();
                        if(newVal > envDataAdded) {
                                JTextField newTf = new JTextField();
                                newTf.setEditable(false);
                                newTf.setColumns(10);

                                JButton newButt = new JButton("Select layers for year " +
                                    Integer.toString(envDataAdded + 1));

                                mpEVPanel.add(newTf, "flowx, cell 0 " + Integer.toString(
                                    envDataAdded) + ", growx");
                                mpEVPanel.add(newButt, "wmin 200, cell 0 " + Integer.toString(
                                    envDataAdded));

                                newButt.addActionListener(new Select_EV(newTf));

                                envDataAdded++;

                                mpEVTF.add(newTf);
                                mpEVBtn.add(newButt);

                                //System.out.println(newVal + "\t" + currSpinnerValue + " add
                                    " + envDataAdded);
                        }
                        else if(newVal < envDataAdded) {
                                mpEVTF.get(envDataAdded - 1).setVisible(false);
                                mpEVBtn.get(envDataAdded - 1).setVisible(false);
                                mpEVPanel.remove(mpEVTF.get(envDataAdded - 1));
                                mpEVPanel.remove(mpEVBtn.get(envDataAdded - 1));

                                mpEVTF.remove(envDataAdded - 1);
```

```
                        mpEVBtn.remove(envDataAdded - 1);

                        envDataAdded--;

                        //System.out.println(newVal + "\t" + currSpinnerValue + "
                            remove" + envDataAdded);
                    }
                    mpEVPanel.validate();
                    mpInputs.validate();
                    mpPanel.validate();

                    envDataAdded = newVal;
                    scrollPane.getVerticalScrollBar().setValue((int)mpEVPanel.
                        getPreferredSize().getHeight());
            }
    }

    public class SP_Filter_Results implements ActionListener {

            @Override
            public void actionPerformed(ActionEvent arg0) {
                    // TODO Auto-generated method stub
                    String filename = "sp";
                    String region = (String) filterBox.getSelectedItem();

                    if(region.equals("- Whole Philippines -")) {
                            ImageDisplayer display = new ImageDisplayer("pictures/
                                predictionsp.png", "sp");
                    }
                    else {
                            for(int i = 0; i < regions.size(); i++) {
                                    if(regions.get(i).getRegionName().equals(region)) {
                                            Filter_Results filter = new Filter_Results(
                                                filename, region, regions.get(i).
                                                getProvinces(), c);
                                            break;
                                    }
                            }
                    }
            }
    }
    public class MP_Filter_Results implements ActionListener {

            @Override
            public void actionPerformed(ActionEvent arg0) {
                    // TODO Auto-generated method stub
                    String filename = "mp";
                    String region = (String) mpFilterBox.getSelectedItem();

                    if(region.equals("- Whole Philippines -")) {
                            //ImageDisplayer display = new ImageDisplayer("pictures/
                                predictionsp.png", "sp");
                            ArrayList<String> imagePaths = new ArrayList<String>();
                            for(int i = 0; i < mpEVTF.size(); i++) {
                                    imagePaths.add("pictures/predictionmp" + i + ".png");
                            }
                            MP_Displayer displayer = new MP_Displayer(imagePaths);
                    }
                    else {
                            for(int i = 0; i < regions.size(); i++) {
                                    if(regions.get(i).getRegionName().equals(region)) {
                                            //Filter_Results filter = new Filter_Results(
                                                filename, region, regions.get(i).
                                                getProvinces(), c);
                                            StringBuilder command = new StringBuilder("");
                                            command.append("filter <- ph[");

                                            for(int j = 0; j < regions.get(i).getProvinces
                                                ().size(); j++) {
                                                if(j == 0) {
                                                        command.append("ph$NAME_1 ==
                                                            \"" + regions.get(i).
                                                            getProvinces().get(j) +
                                                            "\"");
                                                }
                                                else {
                                                        command.append(" | ph$NAME_1
                                                            == \"" + regions.get(i).
                                                            getProvinces().get(j) +
                                                            "\"");
                                                }
                                            }
                                            command.append(",]");

                                            try {
                                                    System.out.println(command.toString())
                                                        ;
                                                    c.eval(command.toString());

                                                    ArrayList<String> imagePaths = new
                                                        ArrayList<String>();

                                                    for(int j = 0; j < mpEVTF.size(); j++)
                                                        {
                                                            command = new StringBuilder
                                                                ("");
                                                            command.append("cr <- crop(
                                                                prediction" + filename + j
                                                                + ", extent(filter), snap
```

```
                                                        = \"out\")");
                                                command.append("\n");
                                                command.append("fr <-
                                                    rasterize(filter, cr)");
                                                command.append("\n");
                                                command.append("filtermp" + j
                                                    + "<- mask(cr, fr)");
                                                command.append("\n");

                                                command.append("png(\"" +
                                                    System.getProperty("user.
                                                    dir").replace('\\', '/') +
                                                    "/pictures/filter"+
                                                    filename + j + ".png\")\n
                                                    ");
                                        command.append("plot(filter"+ filename + j
                                                + ", xlab = 'Longitude', ylab = '
                                                Latitude')\n");
                                        command.append("plot(filter, add = T)\n");
                                        command.append("dev.off()\n");

                                                c.eval(command.toString());
                                                imagePaths.add("pictures/
                                                    filtermp" + j + ".png");
                                        }
                                        MP_Displayer displayer = new
                                            MP_Displayer(imagePaths);

                                        graphPanel.removeAll();

                                    } catch (RserveException e) {
                                        // TODO Auto-generated catch block
                                        e.printStackTrace();
                                        JOptionPane.showMessageDialog(null, "
                                            Error", "There is an error in
                                            project file - assets/regions.txt
                                            ", JOptionPane.ERROR_MESSAGE);
                                    }
                                    break;
                            }
                        }
                        StringBuilder command = new StringBuilder("");
                        command.append("graph <- vector(mode = \"numeric\", length =
                            0)\n");
                        for(int i = 0; i < mpEVTF.size(); i++) {
                            command.append("graph <- append(graph, mean(as.vector(
                                filtermp"+ i +")[!is.na(as.vector(filtermp"+ i +")
                                )]))\n");
                        }
                        command.append("png(\"" + System.getProperty("user.dir").
                            replace('\\', '/') + "/pictures/graph.png\")\n");
                    command.append("plot(seq(1,length(graph)), graph, xlab = 'Year', ylab
                        = 'Likelihood')\n");
                    command.append("lines(seq(1,length(graph)), graph)\n");
                    command.append("dev.off()\n");
                    try {
                        System.out.println(command.toString());
                            c.eval(command.toString());
                        } catch (RserveException e) {
                            // TODO Auto-generated catch block
                            e.printStackTrace();
                        }

                    try {
                            BufferedImage graph = ImageIO.read(new File("pictures/
                                graph.png"));
                            JLabel picLabel = new JLabel(new ImageIcon(graph));

                            graphPanel.removeAll();
                            graphPanel.setLayout(new MigLayout("", "[grow]", "[
                                grow]"));
                            graphPanel.add(picLabel, "cell 0 0, grow, align center
                                ");
                            graphPanel.repaint();
                            graphPanel.revalidate();
                        } catch (IOException e) {
                            // TODO Auto-generated catch block
                            JOptionPane.showMessageDialog(null, "graph.png is
                                missing", "Error", JOptionPane.ERROR_MESSAGE);
                        }
                    }
                }
            }
        }
```

A..2 Java Backend

```
package javaBackend;

import java.io.File;

import javax.swing.JTextField;

public class FileParser {
        public File[] textFieldParser(JTextField tf) {
                String[] filesText = tf.getText().split(";");
                File[] files = new File[filesText.length];
                for(int i = 0; i < filesText.length; i++) {
```

```
                        files[i] = new File(filesText[i]);
                }
                return files;
        }
}

package javaBackend;

import java.util.ArrayList;

import javax.swing.JOptionPane;
import javax.swing.JPanel;

import org.rosuda.REngine.Rserve.RConnection;
import org.rosuda.REngine.Rserve.RserveException;

public class Filter_Results {

        public Filter_Results(String filename, String region, ArrayList<String> provinces,
             RConnection c) {
                StringBuilder command = new StringBuilder("");

                command.append("filter <- ph[");

                for(int i = 0; i < provinces.size(); i++) {
                        if(i == 0) {
                                command.append("ph$NAME_1 == \"" + provinces.get(i) + "\"");
                        }
                        else {
                                command.append(" | ph$NAME_1 == \"" + provinces.get(i) + "\"")
                                     ;
                        }
                }

                command.append(",]");
                try {
                        System.out.println(command.toString());
                        c.eval(command.toString());

                        command = new StringBuilder("");
                        command.append("cr <- crop(prediction" + filename + ", extent(filter),
                              snap = \"out\")");
                        command.append("\n");
                        command.append("fr <- rasterize(filter, cr)");
                        command.append("\n");
                        command.append("filtersp <- mask(cr, fr)");
                        command.append("\n");

                        command.append("png(\"" + System.getProperty("user.dir").replace('\\',
                              '/') + "/pictures/filter"+ filename +".png\")\n");
            command.append("plot(filter"+ filename +", xlab = 'Longitude', ylab = 'Latitude')\
                 n");
            command.append("plot(filter, add = T)\n");
            command.append("dev.off()\n");

                        c.eval(command.toString());

                        ImageDisplayer image = new ImageDisplayer("pictures/filter" + filename
                              + ".png", filename);
                } catch (RserveException e) {
                        // TODO Auto-generated catch block
                        //e.printStackTrace();
                        JOptionPane.showMessageDialog(null, "Error", "There is an error in
                             project file - assets/regions.txt", JOptionPane.ERROR_MESSAGE);
                }

                System.out.println(command.toString());
        }

        public static void main(String[] args) {
//              ArrayList<String> p = new ArrayList<String>();
//              p.add("Prov 1");
//              p.add("Prov 2");
//              String region = "Test Region";
//              String filename = "sp";
//
//              Filter_Results test = new Filter_Results(filename, region, p);
        }

}

package javaBackend;

import java.awt.BorderLayout;
import java.awt.Image;
import java.awt.Window;
import java.io.File;
import java.io.IOException;

import javax.imageio.ImageIO;
import javax.swing.JFrame;
import javax.swing.JOptionPane;

import ui.Main;

public class ImageDisplayer {
        private Image imageFile;
```

```
        public ImageDisplayer(String imagePath, String mode) {
                try {
                        imageFile = ImageIO.read(new File(imagePath));
                } catch (IOException e) {
                        // TODO Auto-generated catch block
                        //e.printStackTrace();
                        JOptionPane.showMessageDialog(null, "Error displaying image");
                }

                TransformingCanvas canvas = new TransformingCanvas(imagePath);

                TranslateHandler translater = new TranslateHandler(canvas);
                canvas.addMouseListener(translater);
                canvas.addMouseMotionListener(translater);

                canvas.addMouseWheelListener(new ScaleHandler(canvas));

                Window[] windows = Window.getWindows();

                if(mode.equals("sp")) {
                        boolean firstWindowChecked = false; //This is a ghetto way of handling
                              this. Please reconsider
                        for(int i = 0; i < windows.length; i++) {
                                if(windows[i].getClass().toString().equals("class javax.swing.
                                     JFrame")) {
                                        if(firstWindowChecked) windows[i].dispose();
                                        else firstWindowChecked = true;
                                }
                        }
                }

                JFrame frame = new JFrame();
                frame.setLayout(new BorderLayout());
                frame.getContentPane().add(canvas, BorderLayout.CENTER);
                frame.setSize(500, 500);
                frame.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
                frame.setVisible(true);

                /*panel.setLayout(new BorderLayout());
                panel.add(canvas, BorderLayout.CENTER);
                canvas.repaint();
                panel.repaint();
                panel.revalidate();*/

        }
}

package javaBackend;

import java.awt.Dimension;
import java.awt.Window;
import java.util.ArrayList;

import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.JScrollPane;

import net.miginfocom.swing.MigLayout;
import tests.GIF_Test;

public class MP_Displayer {

        private JFrame frame;
        private static ArrayList<String> images;

        /**
         * Create the application.
         */
        public MP_Displayer(ArrayList<String> images) {

                this.images = images;

                initialize();
        }

        public static void main(String[] args) {
                ArrayList<String> test = new ArrayList<String>();
                MP_Displayer display = new MP_Displayer(test);
        }

        /**
         * Initialize the contents of the frame.
         */
        private void initialize() {
                Window[] windows = Window.getWindows();
                boolean firstWindowChecked = false; //This is a ghetto way of handling this.
                     Please reconsider
                for(int i = 0; i < windows.length; i++) {
                        if(windows[i].getClass().toString().equals("class javax.swing.JFrame")
                             ) {
                                if(firstWindowChecked) windows[i].dispose();
                                else firstWindowChecked = true;
                        }
                }

                frame = new JFrame();
                frame.setTitle("Dipterocarp Modelling Tool");
                frame.setMinimumSize(new Dimension(500, 500));
                frame.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
```

```
                frame.getContentPane().setLayout(new MigLayout("", "[grow]", "[grow]"));

                JScrollPane scrollPane = new JScrollPane();
                frame.getContentPane().add(scrollPane, "cell 0 0,grow");

                JPanel displayPanel = new JPanel();
                scrollPane.setViewportView(displayPanel);
                displayPanel.setLayout(new MigLayout("", "[grow]", "[grow]"));

                System.out.println(images.size());
                GIF_Test predictions = new GIF_Test(images);
                displayPanel.removeAll();
                displayPanel.add(predictions, "cell 0 0, grow, align center");


                predictions.setVisible(true);
                frame.setVisible(true);
        }

}

package javaBackend;

import java.io.File;

import javax.swing.JOptionPane;
import javax.swing.JPanel;

import org.rosuda.REngine.Rserve.RConnection;
import org.rosuda.REngine.Rserve.RserveException;

public class Predict_Dist {
        private boolean error;
        private RConnection c;

        public boolean getError() {
                return this.error;
        }

        public void getGraph(int numberOfYears, RConnection c) {
                StringBuilder command = new StringBuilder("");
                command.append("graph <- vector(mode = \"numeric\", length = 0)\n");
                for(int i = 0; i < numberOfYears; i++) {
                        command.append("graph <- append(graph, mean(as.vector(predictionmp"+ i
                                +")[!is.na(as.vector(predictionmp"+ i +"))]))\n");
                }
                command.append("png(\"" + System.getProperty("user.dir").replace('\\', '/') +
                     "/pictures/graph.png\")\n");
        command.append("plot(seq(1,length(graph)), graph, xlab = 'Year', ylab = 'Likelihood')\
             n");
        command.append("lines(seq(1,length(graph)), graph)\n");
        command.append("dev.off()\n");
        try {
                System.out.println(command.toString());
                        c.eval(command.toString());
                } catch (RserveException e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                }
        }

        public Predict_Dist(File[] EV, String pathMod, String filename, RConnection c) {
                this.c = c;
                error = false;

                StringBuilder command = new StringBuilder("");
                command.append("model <- readRDS(\"" + pathMod + "\")\n");

                try {
                        System.out.println(command.toString());
                        c.eval(command.toString());
                        command = new StringBuilder("");
                } catch (RserveException e) {
                        // TODO Auto-generated catch block
//                      e.printStackTrace();
                        JOptionPane.showMessageDialog(null, "Invalid model", "Warning",
                             JOptionPane.WARNING_MESSAGE);
                        error = true;
                }
                if(error){}
                else {
                        command.append("files <- as.vector(c(");
                        for(int i = 0; i < EV.length; i++) {
                        if(i == 0) {
                                command.append('"');
                                command.append(EV[i].getAbsolutePath().replace('\\', '/'));
                                command.append('"');
                        }
                        else {
                                command.append(',');
                                command.append('"');
                                command.append(EV[i].getAbsolutePath().replace('\\', '/'));
                                command.append('"');
                        }
                }
                        command.append("))\n");
                        command.append("predictors <- stack(files)\n");
                        command.append("prediction" + filename + " <- predict(model,
                             predictors, progress = 'window')\n");
```

```
                        command.append("cr <- crop(prediction" + filename + ", extent(ph),
                             snap = \"out\")\n");
                        command.append("fr <- rasterize(ph, cr)\n");
                        command.append("prediction" + filename + "<- mask(x=cr, mask=fr)\n");

                        command.append("png(\"" + System.getProperty("user.dir").replace('\\',
                              '/') + "/pictures/prediction"+ filename +".png\")\n");
                command.append("plot(prediction"+ filename +", xlab = 'Longitude', ylab = '
                     Latitude')\n");
                command.append("plot(ph, add = T)\n");
                command.append("dev.off()\n");

//              command.append("png(\"" + System.getProperty("user.dir").replace('\\', '/') + "/
        pictures/weights"+ filename +".png\")\n");
//              command.append("plot(model)\n");
//              command.append("dev.off()\n");

                System.out.println(command.toString());
                try {
                        c.eval(command.toString());
                        } catch (RserveException e) {
                                // TODO Auto-generated catch block
//                              e.printStackTrace();
                                JOptionPane.showMessageDialog(null, "Error while predicting",
                                     "Error", JOptionPane.ERROR_MESSAGE);
                        }


                }
        }
}
```

```
package javaBackend;

import java.util.ArrayList;

public class Region {
        private ArrayList<String> provinces;
        private String regionName;
        public Region(ArrayList<String> provinces, String regionName) {
                this.provinces = provinces;
                this.regionName = regionName;
        }
        public ArrayList<String> getProvinces() {
                return provinces;
        }
        public void setProvinces(ArrayList<String> provinces) {
                this.provinces = provinces;
        }
        public String getRegionName() {
                return regionName;
        }
        public void setRegionName(String regionName) {
                this.regionName = regionName;
        }

}
```

```
package javaBackend;

import java.io.File;
import java.io.FileNotFoundException;
import java.util.ArrayList;
import java.util.Scanner;
import java.util.StringTokenizer;

import javax.swing.JOptionPane;

public class RegionParser {
        public ArrayList<Region> regionParser() {
                File regionFile = new File("assets/regions.txt");
                ArrayList<Region> regions = new ArrayList<Region>();
                try {
                        Scanner sc = new Scanner(regionFile);
                        while(sc.hasNextLine()) {
                                String line = sc.nextLine();


                                StringTokenizer splitLine = new StringTokenizer(line, "|");
                                String regionName = splitLine.nextToken();


                                ArrayList<String> provinces = new ArrayList<String>();
                                while(splitLine.hasMoreTokens()) {
                                        provinces.add(splitLine.nextToken());
                                }
                                regions.add(new Region(provinces, regionName));
                        }
                } catch (FileNotFoundException e) {
                        // TODO Auto-generated catch block
                        JOptionPane.showMessageDialog(null, "Error", "Missing project file -
                             regions.txt", JOptionPane.ERROR_MESSAGE);
                }
                return regions;
        }
        public static void main(String[] args) {
                // TODO Auto-generated method stub
```

```
                RegionParser rp = new RegionParser();
                rp.regionParser();
        }

}


package javaBackend;

import java.io.File;
import java.util.ArrayList;

import javax.swing.JOptionPane;
import javax.swing.JTextField;

import org.rosuda.REngine.Rserve.RConnection;
import org.rosuda.REngine.Rserve.RserveException;

public class Remodelling {
        private RConnection c;
        private JTextField predictors;
        private JTextField datalocs;
        private ArrayList<String> factors;
        private JTextField destTF;

        public Remodelling(RConnection c, JTextField predictors, JTextField datalocs,
             ArrayList<String> factors, JTextField destTF) {
                this.c = c;
                this.predictors = predictors;
                this.datalocs = datalocs;
                this.factors = factors;
                this.destTF = destTF;
        }

        public void model() {
                FileParser parser = new FileParser();
                File[] toStack = parser.textFieldParser(predictors);
                StringBuilder command = new StringBuilder("");

                command.append("locs <- read.csv(\"" + datalocs.getText().replace('\\', '/') +
                      "\")\n");
                command.append("datalocs <- data.frame(locs$longitude, locs$latitude)\n");

                command.append("files <- as.vector(c(");
                for (int i = 0; i < toStack.length; i++) {
                        if (i == 0) {
                                command.append('"');
                                command.append(toStack[i].getAbsolutePath().replace('\\', '/')
                                     );
                                command.append('"');
                        } else {
                                command.append(',');
                                command.append('"');
                                command.append(toStack[i].getAbsolutePath().replace('\\', '/')
                                     );
                                command.append('"');
                        }
                }
                command.append("))\n");
                command.append("predictors <- stack(files)\n");

                if(factors.size() == 0) command.append("model <- maxent(predictors, datalocs")
                     ;
                else {
                        command.append("model <- maxent(predictors, datalocs,factors = ");

                        for(int i = 0; i < factors.size(); i++) {
                                if(i == 0) {
                                        command.append("\"");
                                        command.append(factors.get(i));
                                        command.append("\"");
                                }
                                else {
                                        command.append(", \"");
                                        command.append(factors.get(i));
                                        command.append("\"");
                                }
                        }
                }
                command.append(")\n");
                command.append("saveRDS(model, \"" + destTF.getText().replace('\\', '/') +
                      "\")\n");

                command.append("png(\"" + System.getProperty("user.dir").replace('\\', '/') +
                      "/pictures/weights.png\")\n");
        command.append("plot(model, xlab = 'Variable Contribution', ylab = 'Variable')\n");
        command.append("dev.off()\n");

                System.out.println(command.toString());
                try {
                        c.eval(command.toString());
                } catch (RserveException e) {
                        // TODO Auto-generated catch block
                        JOptionPane.showMessageDialog(null, "Error saving the model. Recheck
                             destination if valid", "Error", JOptionPane.ERROR_MESSAGE);
                }
        }
}
```

```
package javaBackend;

import java.awt.event.MouseWheelEvent;
import java.awt.event.MouseWheelListener;

public class ScaleHandler implements MouseWheelListener {

	private TransformingCanvas canvas;

	public ScaleHandler(TransformingCanvas canvas) {
		this.canvas = canvas;
	}

	public void mouseWheelMoved(MouseWheelEvent e) {
		if(e.getScrollType() == MouseWheelEvent.WHEEL_UNIT_SCROLL) {
			canvas.setScale(canvas.getScale() - (.05 * e.getWheelRotation()));

			// don't cross negative threshold.
			// also, setting scale to 0 has bad effects
			canvas.setScale(Math.max(0, canvas.getScale()));
			canvas.repaint();
		}
	}
}

package javaBackend;

import java.awt.Color;
import java.awt.Graphics;
import java.awt.Graphics2D;
import java.awt.geom.AffineTransform;
import java.awt.image.BufferedImage;
import java.io.File;
import java.io.IOException;

import javax.imageio.ImageIO;
import javax.swing.JComponent;
import javax.swing.JOptionPane;

public class TransformingCanvas extends JComponent {
	private double translateX;
	private double translateY;
	private double scale;
	private String filename;

	TransformingCanvas(String filename) {
		translateX = 0;
		translateY = 0;
		scale = 1;
		this.filename = filename;
		setOpaque(true);
		setDoubleBuffered(true);
	}

	@Override public void paint(Graphics g) {

		AffineTransform tx = new AffineTransform();
		tx.scale(scale, scale);
		tx.translate(translateX, translateY);
		System.out.println("Scale:\t" + scale + " TX:\t" + translateX + "TY:\t" +
		    translateY);
		//System.out.println(translateX + "\t" + translateY);


		Graphics2D ourGraphics = (Graphics2D) g;

		ourGraphics.setColor(Color.WHITE);
		ourGraphics.fillRect(0, 0, getWidth(), getHeight());
		ourGraphics.setTransform(tx);

		//System.out.println("test");

		File image = new File(filename);
		BufferedImage bi;
		try {
			bi = ImageIO.read(image);
			ourGraphics.drawRenderedImage(bi, tx);
		} catch (IOException e) {
			// TODO Auto-generated catch block
			JOptionPane.showMessageDialog(null, "Error displaying image");
		}
	}

	public double getTranslateX() {
		return translateX;
	}

	public void setTranslateX(double translateX) {
		this.translateX = translateX;
	}

	public double getTranslateY() {
		return translateY;
	}

	public void setTranslateY(double translateY) {
		this.translateY = translateY;
	}
```

```
        public double getScale() {
                return scale;
        }

        public void setScale(double scale) {
                this.scale = scale;
        }
}


package javaBackend;

import java.awt.event.MouseEvent;
import java.awt.event.MouseListener;
import java.awt.event.MouseMotionListener;

public class TranslateHandler implements MouseListener,
MouseMotionListener {
        private int lastOffsetX;
        private int lastOffsetY;
        private TransformingCanvas canvas;

        public TranslateHandler(TransformingCanvas canvas) {
                this.canvas = canvas;
        }

        public void mousePressed(MouseEvent e) {
                // capture starting point
                lastOffsetX = e.getX();
                lastOffsetY = e.getY();
        }

        public void mouseDragged(MouseEvent e) {
                // new x and y are defined by current mouse location subtracted
                // by previously processed mouse location
                int newX = e.getX() - lastOffsetX;
                int newY = e.getY() - lastOffsetY;


                // increment last offset to last processed by drag event.
                lastOffsetX += newX;
                lastOffsetY += newY;

                // update the canvas locations
                canvas.setTranslateX(canvas.getTranslateX() + newX);
                canvas.setTranslateY(canvas.getTranslateY() + newY);

                // schedule a repaint.
                canvas.repaint();
        }

        public void mouseClicked(MouseEvent e) {}
        public void mouseEntered(MouseEvent e) {}
        public void mouseExited(MouseEvent e) {}
        public void mouseMoved(MouseEvent e) {}
        public void mouseReleased(MouseEvent e) {}
}


package tests;

import java.awt.BorderLayout;
import java.awt.Graphics;
import java.awt.MediaTracker;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.util.ArrayList;

import javax.swing.ImageIcon;
import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.Timer;

import net.miginfocom.swing.MigLayout;

public class GIF_Test extends JPanel implements ActionListener {
        ImageIcon images[];
        int totalImages, currentImage = 0, animationDelay = 300;
        Timer animationTimer;

        public GIF_Test(ArrayList<String> paths) {
                totalImages = paths.size();
                images = new ImageIcon[totalImages];
                for (int i = 0; i < images.length; ++i)
                        images[i] = new ImageIcon(paths.get(i));
                startAnimation();
        }

        public void paintComponent(Graphics g) {
                super.paintComponent(g);
                if (images[currentImage].getImageLoadStatus() == MediaTracker.COMPLETE) {
                        int x = (this.getWidth() - images[currentImage].getIconWidth()) / 2;
                    int y = (this.getHeight() - images[currentImage].getIconHeight()) / 2;
                        images[currentImage].paintIcon(this, g, x, y);
                        currentImage = (currentImage + 1) % totalImages;
                }
        }

        public void actionPerformed(ActionEvent e) {
                repaint();
```

```
        }

        public void startAnimation() {
                if (animationTimer == null) {
                        currentImage = 0;
                        animationTimer = new Timer(animationDelay, this);
                        animationTimer.start();
                } else if (!animationTimer.isRunning())
                        animationTimer.restart();
        }

        public void stopAnimation() {
                animationTimer.stop();
        }

        public static void main(String args[]) {
                ArrayList<String> pics = new ArrayList<String>();
                pics.add("pictures/predictionmp0.png");
                pics.add("pictures/predictionmp1.png");
                GIF_Test anim = new GIF_Test(pics);
                JFrame app = new JFrame("Animator test");
                app.setLayout(new MigLayout("", "[grow]", "[grow]"));
                app.add(anim, "cell 0 0");
                app.setSize(300, 300);
                app.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
                app.setVisible(true);
        }
}
```

A..3 R Backend

```
package rBackend;

import java.io.File;

import javax.swing.JOptionPane;

import org.rosuda.REngine.Rserve.RConnection;
import org.rosuda.REngine.Rserve.RserveException;

public class DatalocsValidator {
        private File datalocs;
        private RConnection c;

        public DatalocsValidator(File datalocs, RConnection c) {
                this.datalocs = datalocs;
                this.c = c;
        }

        public boolean areDatalocsValid() {
                StringBuilder command;

                try {
                        command = new StringBuilder("");

                        command.append("locs <- read.csv(\"" + datalocs.getAbsolutePath().
                             replace('\\', '/') + "\")\n");
                        command.append("datalocs <- data.frame(locs$longitude, locs$latitude)\
                             n");

                        c.eval(command.toString());
                }
                catch(RserveException rs) {
                        JOptionPane.showMessageDialog(null, "There is a problem with input
                             data localities. Please recheck the file.", "Error", JOptionPane.
                             ERROR_MESSAGE);
                        return false;
                }

                return true;
        }
}
```

```
package rBackend;

import java.io.File;

import javax.swing.JOptionPane;

import org.rosuda.REngine.Rserve.RConnection;
import org.rosuda.REngine.Rserve.RserveException;

public class EnviVarValidator {
        private RConnection c;
        private File[] enviVars;

        public EnviVarValidator(RConnection c, File[] enviVars) {
                this.c = c;
                this.enviVars = enviVars;
        }

        public boolean areEnviVarValid() {
                StringBuilder command;
                for(int i = 0; i < enviVars.length; i++) {
                        try {
                                command = new StringBuilder("stack(\"" + enviVars[i].
                                     getAbsolutePath() + "\")");
```

```
                        c.eval(command.toString().replace('\\', '/'));
                }
                catch(RserveException rs) {
                        JOptionPane.showMessageDialog(null, "Invalid file: " +
                             enviVars[i].getName(), "Error", JOptionPane.ERROR_MESSAGE)
                             ;
                        return false;
                }
        }
        try {
                command = new StringBuilder("files <- as.vector(c(");

        for(int i = 0; i < enviVars.length; i++) {
                if(i == 0) {
                        command.append('"');
                        command.append(enviVars[i].getAbsolutePath().replace('\\',
                             '/'));
                        command.append('"');
                }
                else {
                        command.append(',');
                        command.append('"');
                        command.append(enviVars[i].getAbsolutePath().replace('\\',
                             '/'));
                        command.append('"');
                }
        }
        command.append("))\n");
        command.append("stack(files)");
                c.eval(command.toString());
        }
        catch(RserveException rs) {
                JOptionPane.showMessageDialog(null, "Error combining the environmental
                     variables. Check if extent of each file are same", "Error",
                    JOptionPane.ERROR_MESSAGE);
                return false;
        }

        return true;
    }
    public static void main(String[] args) throws RserveException {
        RConnector s = new RConnector();
        RConnection c = s.getRConnection();
        File[] files = new File[2];
        files[0] = new File("C:/Users/adria/Desktop/test.png");
        files[1] = new File("C:/Users/adria/Desktop/test.tif");

        EnviVarValidator test = new EnviVarValidator(c, files);
        System.out.println(test.areEnviVarValid());
    }
}

package rBackend;

import java.io.IOException;

import javax.swing.JOptionPane;

import org.rosuda.REngine.Rserve.RConnection;
import org.rosuda.REngine.Rserve.RserveException;

public class RConnector {
    private RConnection c;

    public RConnector() {
        startRConnection();
        loadLibraries();
    }

    public void loadLibraries() {
        try {
            StringBuilder command = new StringBuilder("");
            command.append("library(dismo)");
            command.append("\n");
            command.append("library(sdm)");
            command.append("\n");
            command.append("library(raster)");
            command.append("\n");
            command.append("library(rgdal)");
            command.append("\n");
            command.append("ph <- readOGR(\"" + System.getProperty("user.dir").
                 replace('\\', '/') + "/PHL_adm_shp/PHL_adm1.shp\")\n");
            c.eval(command.toString());
            System.out.println("Libraries loaded");
        } catch (RserveException e) {
            // TODO Auto-generated catch block
            //e.printStackTrace();
            System.out.println("Ui Rserve Exception");
        }
    }

    public void startRConnection() {
        try {
            Process pro = null;
            ProcessBuilder pb = new ProcessBuilder("RScript","assets/InitRserve.
                 txt");

            pro = pb.start();
            pro.waitFor();
```

```
                        pro.destroy();
                }
                catch(IOException io) {
                        System.out.println("IOException starting Rserve");
                } catch (InterruptedException e) {
                        // TODO Auto-generated catch block
                        System.out.println("Interrupted starting Rserve");
                }
                while(c == null) {
                        try {
                                c = new RConnection("localhost");

                                System.out.println("Connected to Rserve");
                        } catch (RserveException e) {
                                // TODO Auto-generated catch block
                                //e.printStackTrace();
                                System.out.println("Ui Rserve Exception");
                        }
                }
        }
        public RConnection getRConnection() {
                return this.c;
        }
}
```

A..4 Component Actions

```
package componentActions;

import javax.swing.JButton;
import javax.swing.JCheckBox;
import javax.swing.JTextField;
import javax.swing.event.ChangeEvent;
import javax.swing.event.ChangeListener;

public class Enable_Own_Model implements ChangeListener{
        private JCheckBox chckbx;
        private JButton btn;


        public Enable_Own_Model(JCheckBox chckbx, JButton btn) {
                this.chckbx = chckbx;
                this.btn = btn;
        }

        @Override
        public void stateChanged(ChangeEvent arg0) {
                // TODO Auto-generated method stub
                if(!chckbx.isSelected()) {
                        btn.setEnabled(true);
                }
                else {
                        btn.setEnabled(false);
                }
        }


}

package componentActions;

import java.awt.BorderLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.image.BufferedImage;
import java.io.File;
import java.io.IOException;
import java.util.ArrayList;

import javax.imageio.ImageIO;
import javax.swing.ImageIcon;
import javax.swing.JButton;
import javax.swing.JCheckBox;
import javax.swing.JComboBox;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JPanel;
import javax.swing.JTextField;

import org.rosuda.REngine.Rserve.RConnection;

import javaBackend.FileParser;
import javaBackend.MP_Displayer;
import javaBackend.Predict_Dist;
import net.miginfocom.swing.MigLayout;

public class MP_Predict implements ActionListener {
        private JTextField tfMod;
        private ArrayList<JTextField> tfEV;
        private JCheckBox defaultModel;
        private RConnection c;
        private JComboBox filterBox;
        private JButton display;
        private Predict_Dist predict;
        private JPanel graphPanel;

        public MP_Predict(JTextField tfMod, ArrayList<JTextField> tfEV, JCheckBox defaultModel
             , RConnection c, JComboBox filterBox, JButton display, JPanel graphPanel) {
```

```
        this.tfMod = tfMod;
        this.tfEV = tfEV;
        this.defaultModel = defaultModel;
        this.c = c;
        this.filterBox = filterBox;
        this.display = display;
        this.graphPanel = graphPanel;
}

public void actionPerformed(ActionEvent arg0) {
        // TODO Auto-generated method stub
        boolean error = false;
        for(JTextField tf : tfEV) {
                if(tf.getText().equals("")) {
                        JOptionPane.showMessageDialog(null, "No environmental
                             variables selected for one of the years", "Warning",
                             JOptionPane.WARNING_MESSAGE);
                        error = true;
                }
        }
        if(!error) {
                error = false;
                if(defaultModel.isSelected()) {
                        ArrayList<String> imagePaths = new ArrayList<String>();
                        for(int i = 0; i < tfEV.size(); i++) {
                                FileParser parser = new FileParser();
                                File[] EV = parser.textFieldParser(tfEV.get(i));
                                predict = new Predict_Dist(EV, System.getProperty("
                                     user.dir").replace('\\', '/') +"/model/model.rds",
                                      "mp" + i, c);
                                if(!error && predict.getError()) error = true;
                                imagePaths.add("pictures/predictionmp" + i + ".png");
                        }
                        if(error){
                                JOptionPane.showMessageDialog(null, "There's a problem
                                      with some of your environmental variables. Please
                                      recheck", "Error", JOptionPane.ERROR_MESSAGE);
                        }
                        else {
                                MP_Displayer displayer = new MP_Displayer(imagePaths);
                                predict.getGraph(tfEV.size(), c);
                                try {
                                        BufferedImage graph = ImageIO.read(new File("
                                             pictures/graph.png"));
                                        JLabel picLabel = new JLabel(new ImageIcon(
                                             graph));

                                        graphPanel.removeAll();
                                        graphPanel.setLayout(new MigLayout("", "[grow
                                             ]", "[grow]"));
                                        graphPanel.add(picLabel, "cell 0 0, grow,
                                             align center");
                                        graphPanel.repaint();
                                        graphPanel.revalidate();

                                        filterBox.setEnabled(true);
                                        display.setEnabled(true);
                                } catch (IOException e) {
                                        // TODO Auto-generated catch block
                                        JOptionPane.showMessageDialog(null, "graph.png
                                              is missing", "Error", JOptionPane.
                                             ERROR_MESSAGE);
                                }
                        }

                }
                else {
                        if(tfMod.getText().equals("")) {
                                JOptionPane.showMessageDialog(null, "No model selected
                                     ", "Warning", JOptionPane.WARNING_MESSAGE);
                        }
                        else {
                                //Predict
                                ArrayList<String> imagePaths = new ArrayList<String>()
                                     ;
                                error = false;
                                for(int i = 0; i < tfEV.size(); i++) {
                                        FileParser parser = new FileParser();
                                        File[] EV = parser.textFieldParser(tfEV.get(i)
                                             );
                                        predict = new Predict_Dist(EV, tfMod.getText()
                                             .replace('\\', '/'), "mp" + i, c);
                                        if(!error && predict.getError()) error = true;
                                        imagePaths.add("pictures/predictionmp" + i +
                                             ".png");
                                }
                                if(error){
                                        JOptionPane.showMessageDialog(null, "There's a
                                              problem with some of your environmental
                                             variables. Please recheck", "Error",
                                             JOptionPane.ERROR_MESSAGE);
                                }
                                else {
                                        MP_Displayer displayer = new MP_Displayer(
                                             imagePaths);
                                        predict.getGraph(tfEV.size(), c);
                                        try {
                                                BufferedImage graph = ImageIO.read(new
                                                     File("pictures/graph.png"));
```

```
                                                        JLabel picLabel = new JLabel(new
                                                            ImageIcon(graph));

                                                        graphPanel.removeAll();
                                                        graphPanel.setLayout(new MigLayout("",
                                                              "[grow]", "[grow]"));
                                                        graphPanel.add(picLabel, "cell 0 0,
                                                            grow, align center");
                                                        graphPanel.repaint();
                                                        graphPanel.revalidate();

                                                        filterBox.setEnabled(true);
                                                        display.setEnabled(true);
                                                } catch (IOException e) {
                                                        // TODO Auto-generated catch block
                                                        JOptionPane.showMessageDialog(null, "
                                                            graph.png is missing", "Error",
                                                            JOptionPane.ERROR_MESSAGE);
                                                }
                                        }
                                }
                        }
                }
        }
}

package componentActions;

import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.io.File;
import java.util.ArrayList;
import java.util.StringTokenizer;

import javax.swing.JButton;
import javax.swing.JOptionPane;
import javax.swing.JTextField;

import org.rosuda.REngine.Rserve.RConnection;

import javaBackend.FileParser;
import javaBackend.Remodelling;
import rBackend.DatalocsValidator;
import rBackend.EnviVarValidator;

public class Remodel_Prep implements ActionListener {
        private RConnection c;
        private JTextField predictors;
        private JTextField datalocs;
        private JTextField factors;
        private JTextField destTF;
        private JButton showContrib;

        public Remodel_Prep(RConnection c, JTextField predictors, JTextField datalocs,
            JTextField factors, JTextField destTF, JButton showContrib) {
                this.c = c;
                this.predictors = predictors;
                this.datalocs = datalocs;
                this.factors = factors;
                this.destTF = destTF;
                this.showContrib = showContrib;
        }
        public static void main(String[] args) {
                String test = "12345";
                System.out.println(test.indexOf("a"));
        }
        @Override
        public void actionPerformed(ActionEvent arg0) {
                // TODO Auto-generated method stub
                boolean predictorError = false;
                boolean locsError = false;
                boolean factorError = false;

                FileParser parser = new FileParser();
                File[] filePredictors = parser.textFieldParser(predictors);

                EnviVarValidator evValidator = new EnviVarValidator(c, filePredictors);
                predictorError = !evValidator.areEnviVarValid();

                DatalocsValidator dlValidator = new DatalocsValidator(new File(datalocs.
                    getText()), c);
                locsError = !dlValidator.areDatalocsValid();

                StringTokenizer st = new StringTokenizer(factors.getText(), ",");
                ArrayList<String> factorString = new ArrayList<String>();
                while(st.hasMoreTokens()) {
                        factorString.add(st.nextToken());
                }
                for(int i = 0; i < factorString.size(); i++) {
                        boolean hasMatch = false;
                        for(int j = 0; j < filePredictors.length; j++) {
                                if(filePredictors[j].getAbsolutePath().indexOf(factorString.
                                    get(i)) != -1) hasMatch = true;
                        }
                        if(!hasMatch) {
                                factorError = true;
                                break;
                        }
                }
```

```
                if(factorError) {
                        JOptionPane.showMessageDialog(null, "A factor doesn't match any of the
                                predictors", "Error", JOptionPane.ERROR_MESSAGE);
                }
                System.out.println(predictorError + "\t" + locsError + "\t" + factorError);
                if(!predictorError && !locsError && !factorError) {
                        Remodelling model = new Remodelling(c, predictors, datalocs,
                                factorString, destTF);
                        model.model();
                        showContrib.setEnabled(true);
                        System.out.println("Yey?");
                }
        }
}
```

```
package componentActions;

import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.io.File;

import javax.swing.JFileChooser;
import javax.swing.JOptionPane;
import javax.swing.JTextField;

public class Select_EV implements ActionListener{

        private JTextField tf;

        public Select_EV(JTextField tf) {
                this.tf = tf;
        }

        @Override
        public void actionPerformed(ActionEvent e) {
                // TODO Auto-generated method stub
                String choosertitle = "Select environmental variables";

                JFileChooser chooser = new JFileChooser();
            chooser.setCurrentDirectory(new java.io.File("."));
            chooser.setDialogTitle(choosertitle);
            chooser.setMultiSelectionEnabled(true);

            chooser.setAcceptAllFileFilterUsed(false);

            if (chooser.showOpenDialog(null) == JFileChooser.APPROVE_OPTION) {
                StringBuilder filenames = new StringBuilder("");
                File[] files = chooser.getSelectedFiles();
                for(File file : files) {
                        //filenames.append('"');
                        filenames.append(file.getAbsolutePath());
                        //filenames.append('"');
                        filenames.append(';');
                }
                tf.setText(filenames.toString());
            }
            else {
                JOptionPane.showMessageDialog(null, "No environmental variables selected");
            }
        }

}
```

```
package componentActions;

import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

import javax.swing.JFileChooser;
import javax.swing.JOptionPane;
import javax.swing.JTextField;
import javax.swing.filechooser.FileNameExtensionFilter;

public class Select_Model implements ActionListener {
        private JTextField tf = new JTextField();

        public Select_Model(JTextField textfield) {
                tf = textfield;
        }

        @Override
        public void actionPerformed(ActionEvent e) {
                // TODO Auto-generated method stub
                String choosertitle = "Select your model";

                JFileChooser chooser = new JFileChooser();
            chooser.setCurrentDirectory(new java.io.File("."));
            chooser.setDialogTitle(choosertitle);

            FileNameExtensionFilter filter = new FileNameExtensionFilter("*.rds", "rds");
            chooser.setFileFilter(filter);


            chooser.setAcceptAllFileFilterUsed(false);

            if (chooser.showOpenDialog(null) == JFileChooser.APPROVE_OPTION) {
                tf.setText(chooser.getSelectedFile().getAbsolutePath());
```

```
            }
            else {
                JOptionPane.showMessageDialog(null, "No model selected");
            }
        }

}


package componentActions;

import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

import javaBackend.ImageDisplayer;

public class SP_LastPred implements ActionListener {

        @Override
        public void actionPerformed(ActionEvent arg0) {
                // TODO Auto-generated method stub
                //ImageDisplayer prediction = new ImageDisplayer("pictures/predictionsp.png");
        }

}


package componentActions;

import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.io.File;

import javax.swing.JButton;
import javax.swing.JCheckBox;
import javax.swing.JComboBox;
import javax.swing.JOptionPane;
import javax.swing.JPanel;
import javax.swing.JTextField;

import org.rosuda.REngine.Rserve.RConnection;

import javaBackend.FileParser;
import javaBackend.ImageDisplayer;
import javaBackend.Predict_Dist;

public class SP_Predict implements ActionListener{
        private JTextField tfMod;
        private JTextField tfEV;
        private JCheckBox defaultModel;
        private RConnection c;
        private JButton filterBtn;
        private JComboBox filterBox;


        public SP_Predict(JTextField tfMod, JTextField tfEV,  JCheckBox defaultModel,
             RConnection c, JButton filterBtn, JComboBox filterBox) {
                this.tfMod = tfMod;
                this.tfEV = tfEV;
                this.defaultModel = defaultModel;
                this.c = c;
                this.filterBox = filterBox;
                this.filterBtn = filterBtn;
        }

        @Override
        public void actionPerformed(ActionEvent arg0) {
                // TODO Auto-generated method stub
                if(tfEV.getText().equals("")) {
                        JOptionPane.showMessageDialog(null, "No environmental variables
                             selected", "Warning", JOptionPane.WARNING_MESSAGE);
                }
                else if(defaultModel.isSelected()) {
                        //Predict
                        filterBox.setEnabled(true);
                        filterBtn.setEnabled(true);
                        FileParser parser = new FileParser();
                        File[] EV = parser.textFieldParser(tfEV);
                        Predict_Dist predict = new Predict_Dist(EV, System.getProperty("user.
                             dir").replace('\\', '/') +"/model/model.rds", "sp", c);
                        if(!predict.getError()) {
                                ImageDisplayer image = new ImageDisplayer("pictures/prediction
                                     " + "sp" + ".png", "sp");
                        }
                }
                else {
                        if(tfMod.getText().equals("")) {
                                JOptionPane.showMessageDialog(null, "No model selected", "
                                     Warning", JOptionPane.WARNING_MESSAGE);
                        }
                        else {
                                //Predict
                                filterBox.setEnabled(true);
                                filterBtn.setEnabled(true);
                                FileParser parser = new FileParser();
                                File[] EV = parser.textFieldParser(tfEV);
                                Predict_Dist predict = new Predict_Dist(EV, tfMod.getText().
                                     replace('\\', '/'), "sp", c);
                                if(!predict.getError()) {
```

```
                                        ImageDisplayer image = new ImageDisplayer("pictures/
                                            prediction" + "sp" + ".png", "sp");
                                    }
                                }
                            }

                        }
                    }
```

XI. Acknowledgement

There is no strict format or language for the acknowledgement. You can write this section in whatever creative way to want to but the language should be limited to English or Filipino.