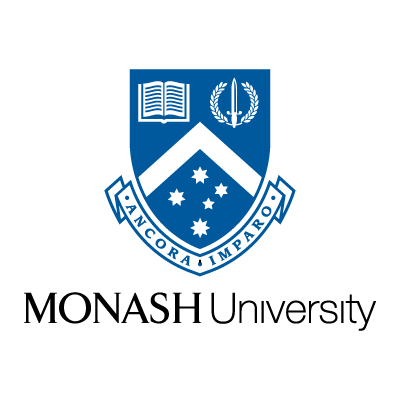
****

**Monash University**

**FIT 3164 Test Report**

**Group 13**

**Members:**

**Haoying Zhang 27791912**

**Haoyang Zheng 28764838**

**Xiaoran Gui 27178234**

[**Data preparation Rscript**](#_xdhtgczhtvb8) **3**

[**Shiny UI Testing**](#_mj23jalzvhy1) **17**

[**Shiny Error Testing**](#_itodlgmtdqrk) **20**

# Data preparation Rscript

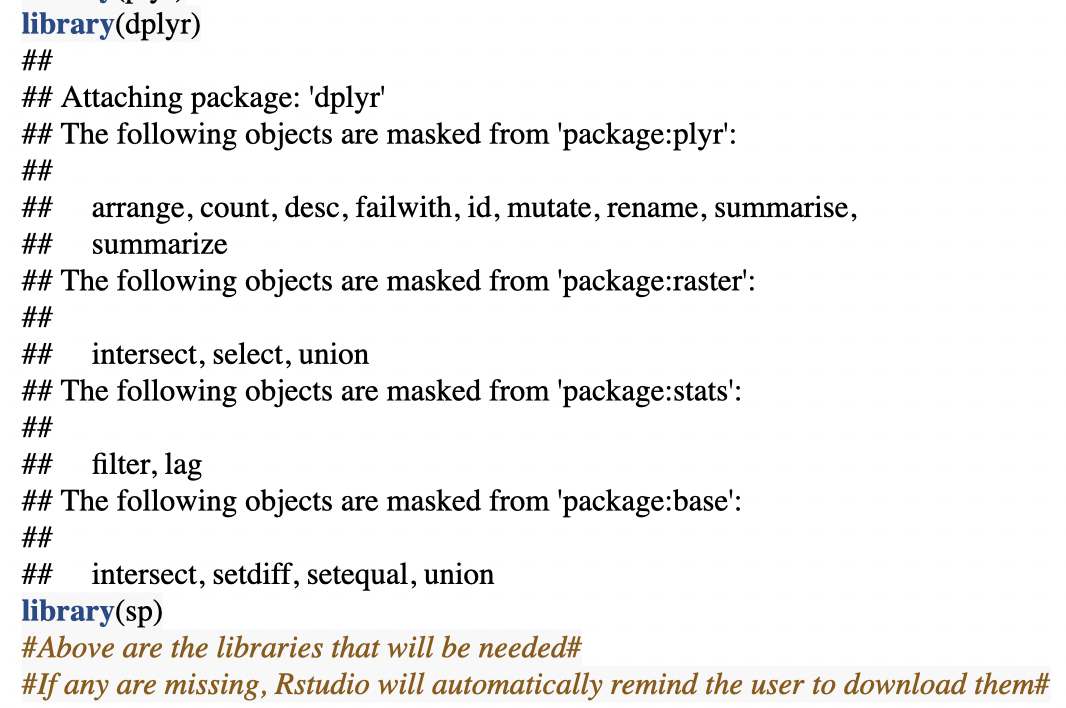
**For testing of the Rscript, all codes are manually ran and tested to check if any errors arises.**

##DOUBLE HASHTAG## = System output

**Load libraries:**

Loading the required libraries



****

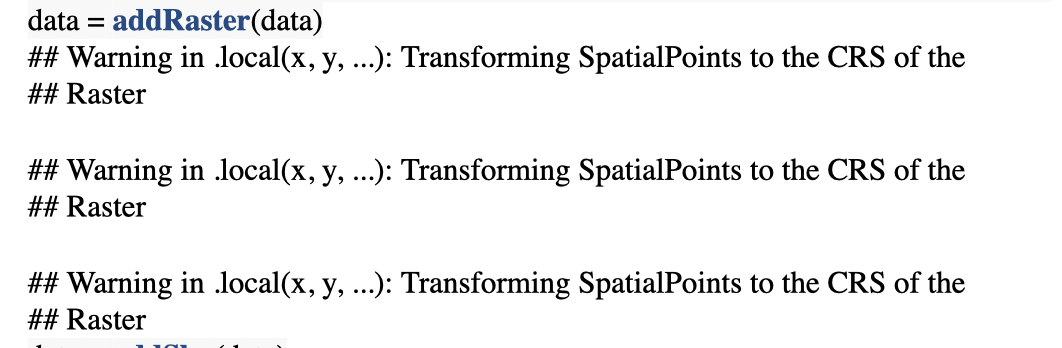
No error has occurred and the libraries are all loaded into the environment in R.

**Testing of addRaster function**

Part of the function code:

****

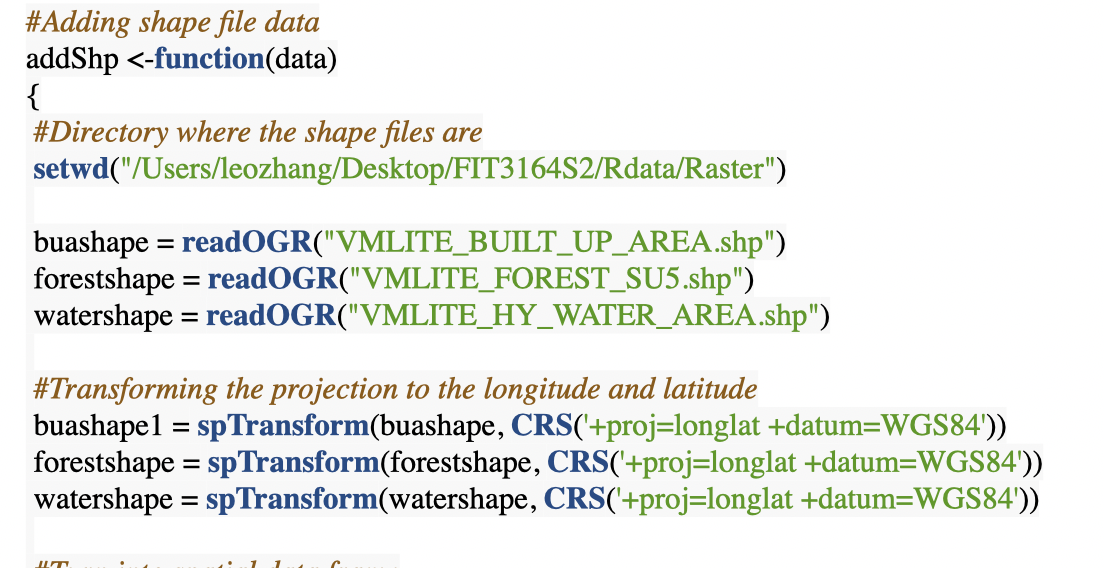
**Test result:**

**Expected the function to run successful**

Testing successful. Warning are given because we are changing the extent of the VBA file to the same as raster files and Rstudio likes to notify the users that we are messing with the projection.

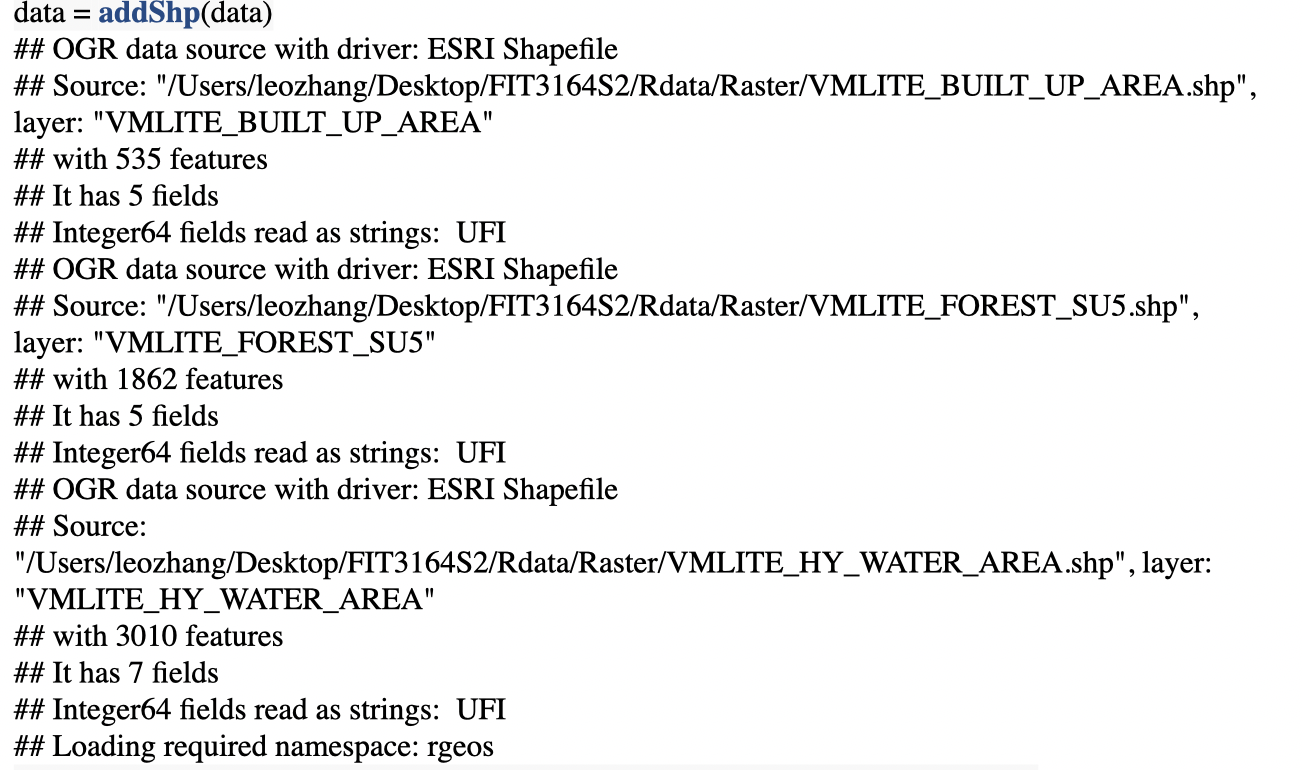
**Testing of addShp function**

Part of the addShp function code:

****

**Result:**

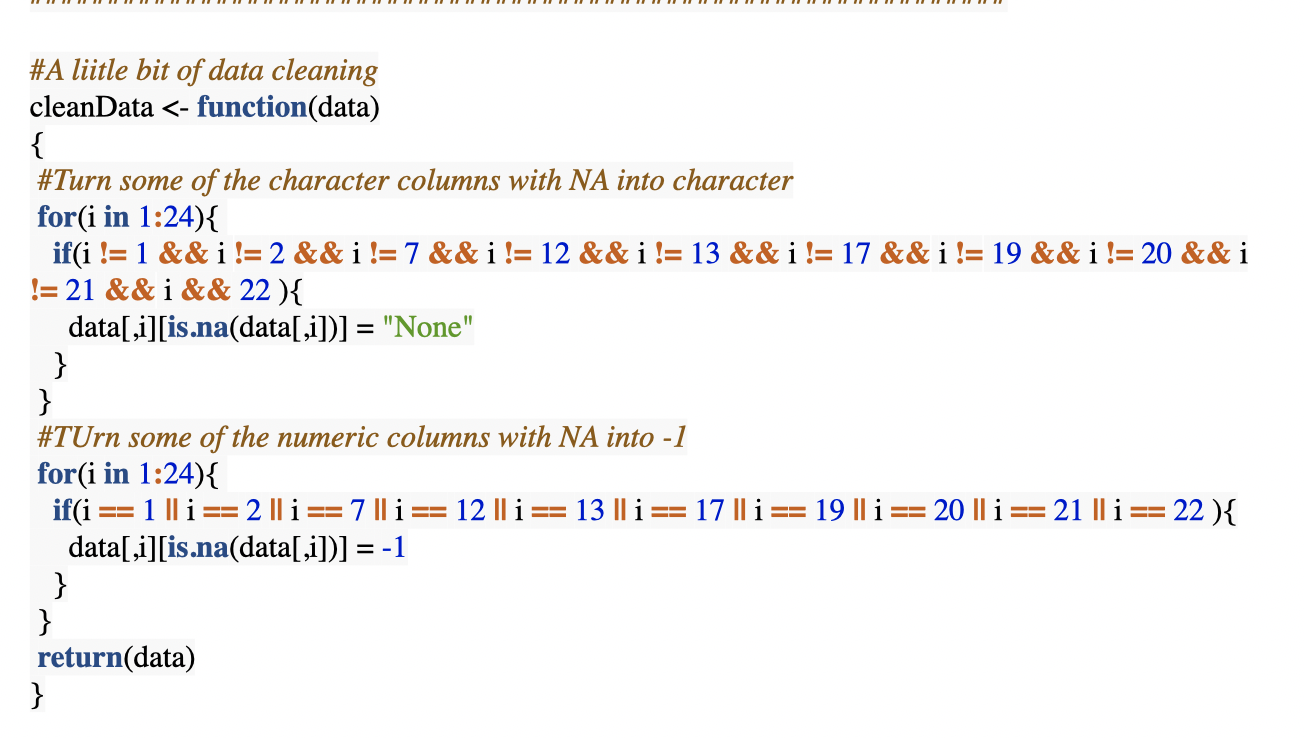
**Expected the function to run without errors**

****

Everything was done perfectly, no error was returned.

**Testing of cleanData function**

Function code:

****

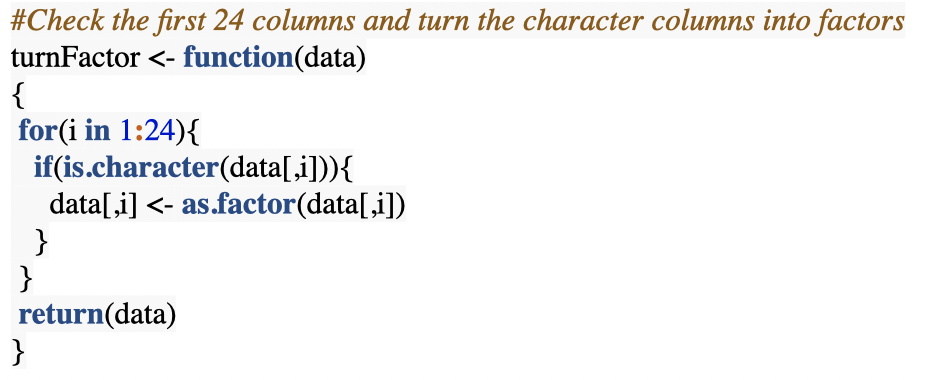
**Testing:**

**The data should now be cleaned after running the running the function.**

No error was returned, the code executed perfectly.

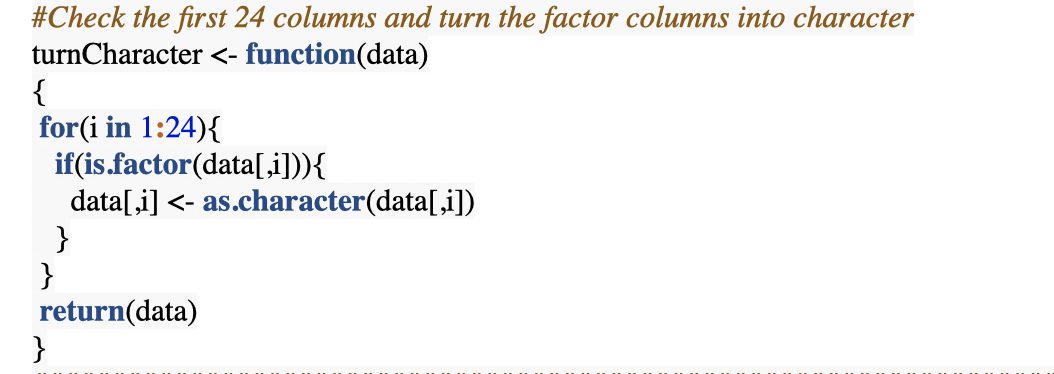
**Testing of clean turnFactor function**

**Function code:**

****

**Testing of turnCharacter function**

**Function code:**

****

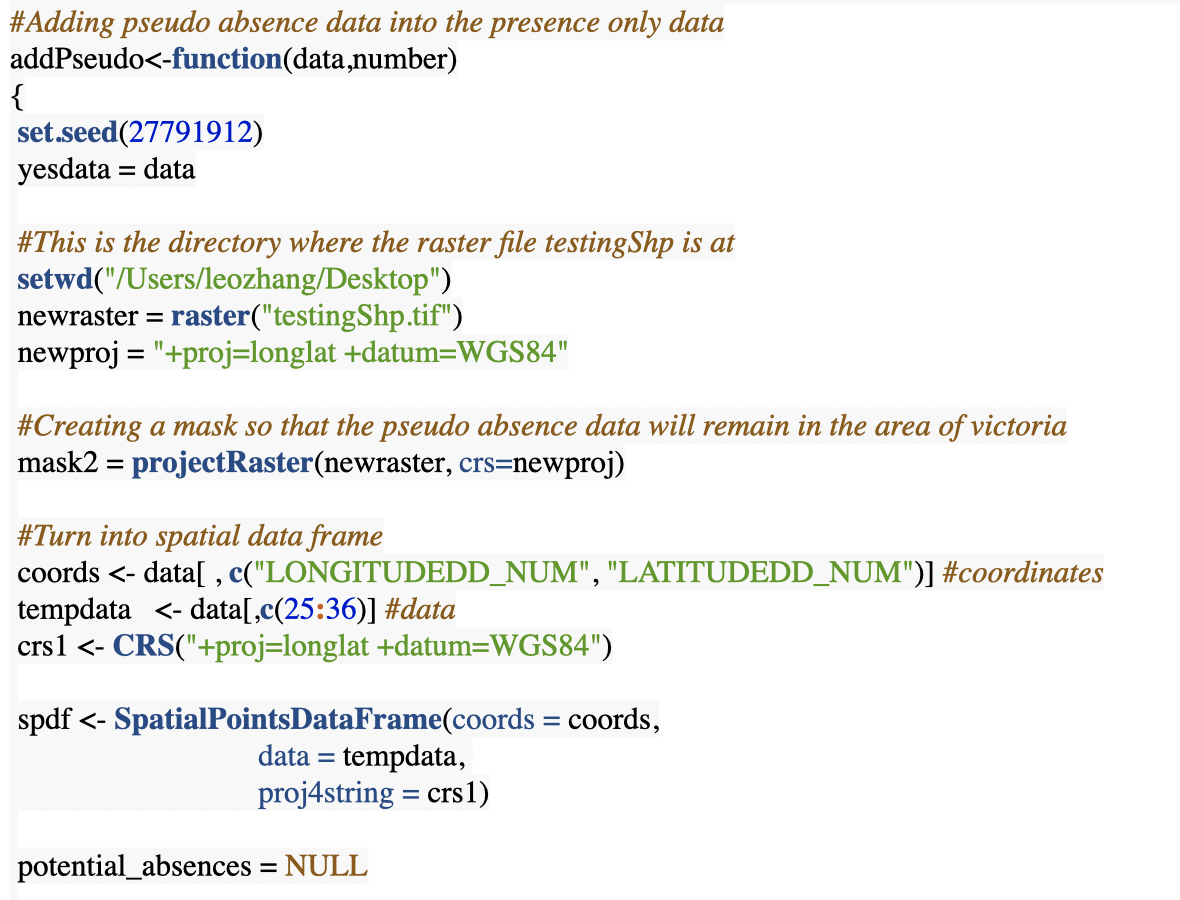
**Result:**

****

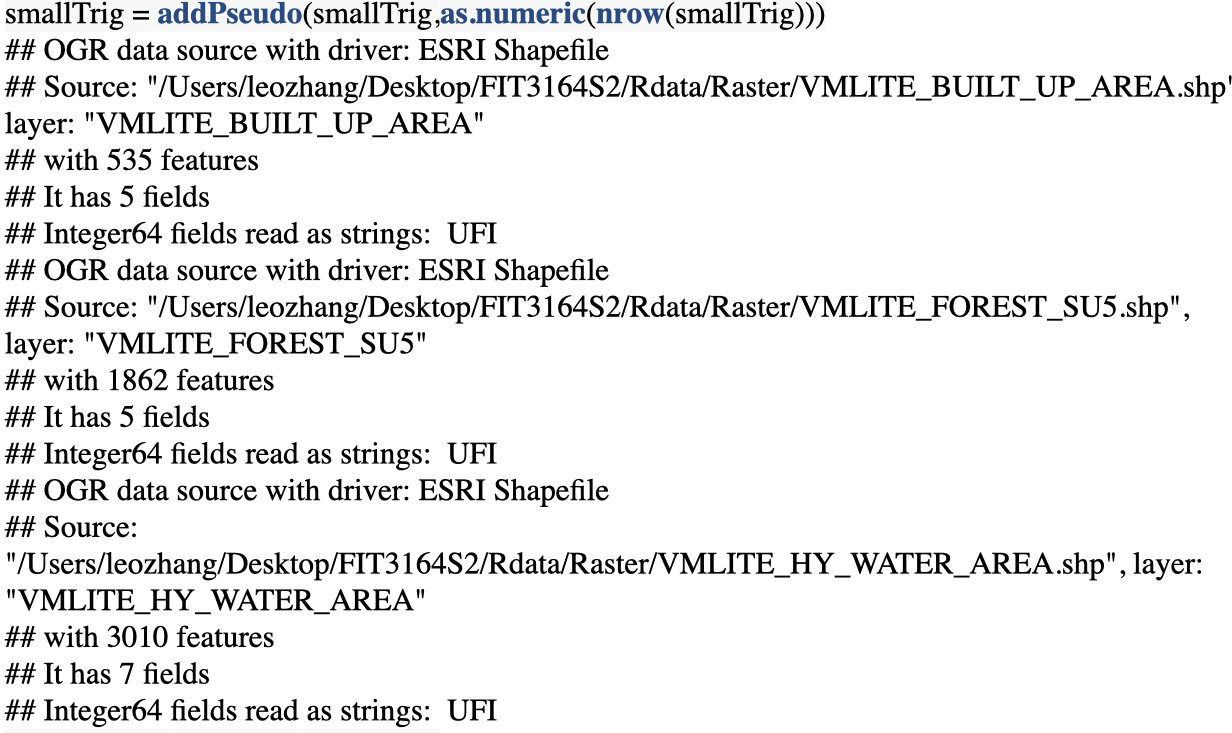
The function worked as expected.

**Testing addPseudo function:**

**Function code:**

****

**Using the species to test the function, if it works for this species it will work for all species because it will use the same type of input.**

****

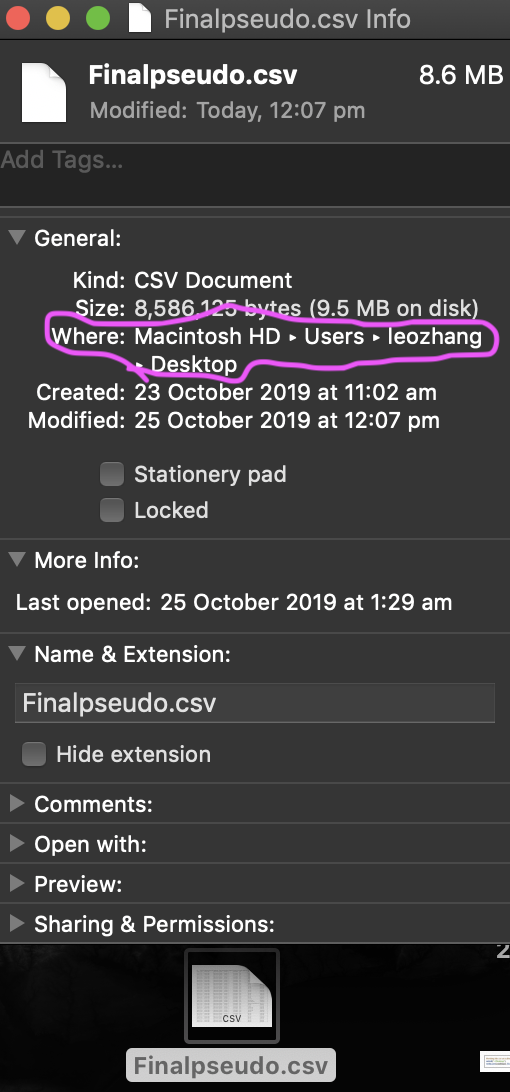
No error was returned

Other species were not documented here because there are no errors as we can see above.

**Testing write file part of the preparation stage.**

****

No error returned and file was created at the Desktop.

****

**Reading the file to test if the file works**

****

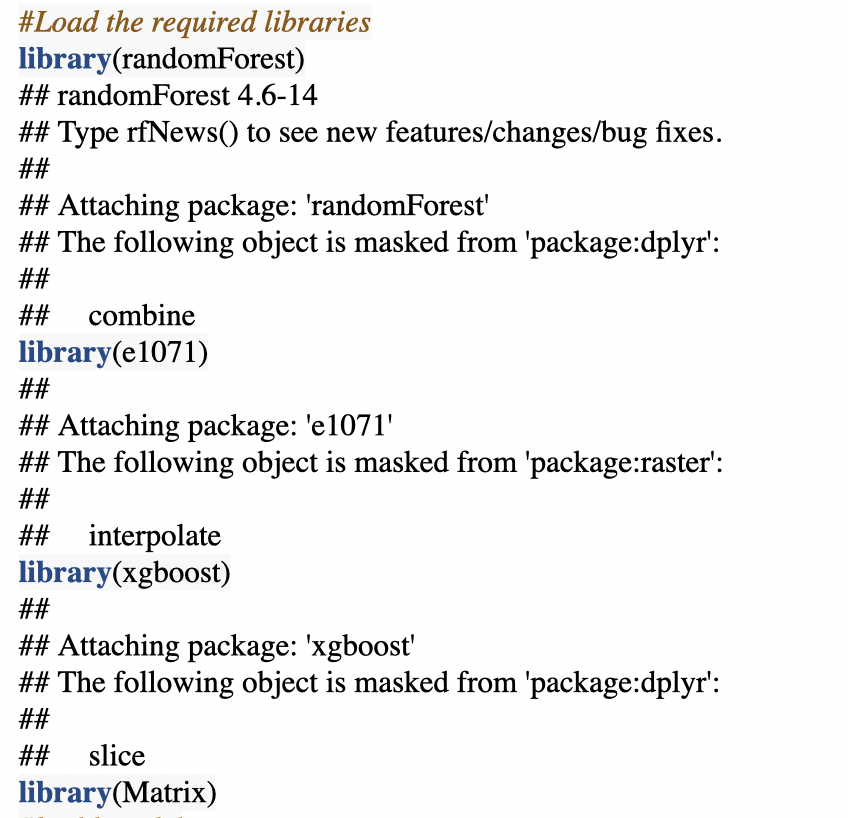
No error returned and data frame created.

**Splitting the species for model generation:**

****

No error returned

**Load new libraries that are needed**

****

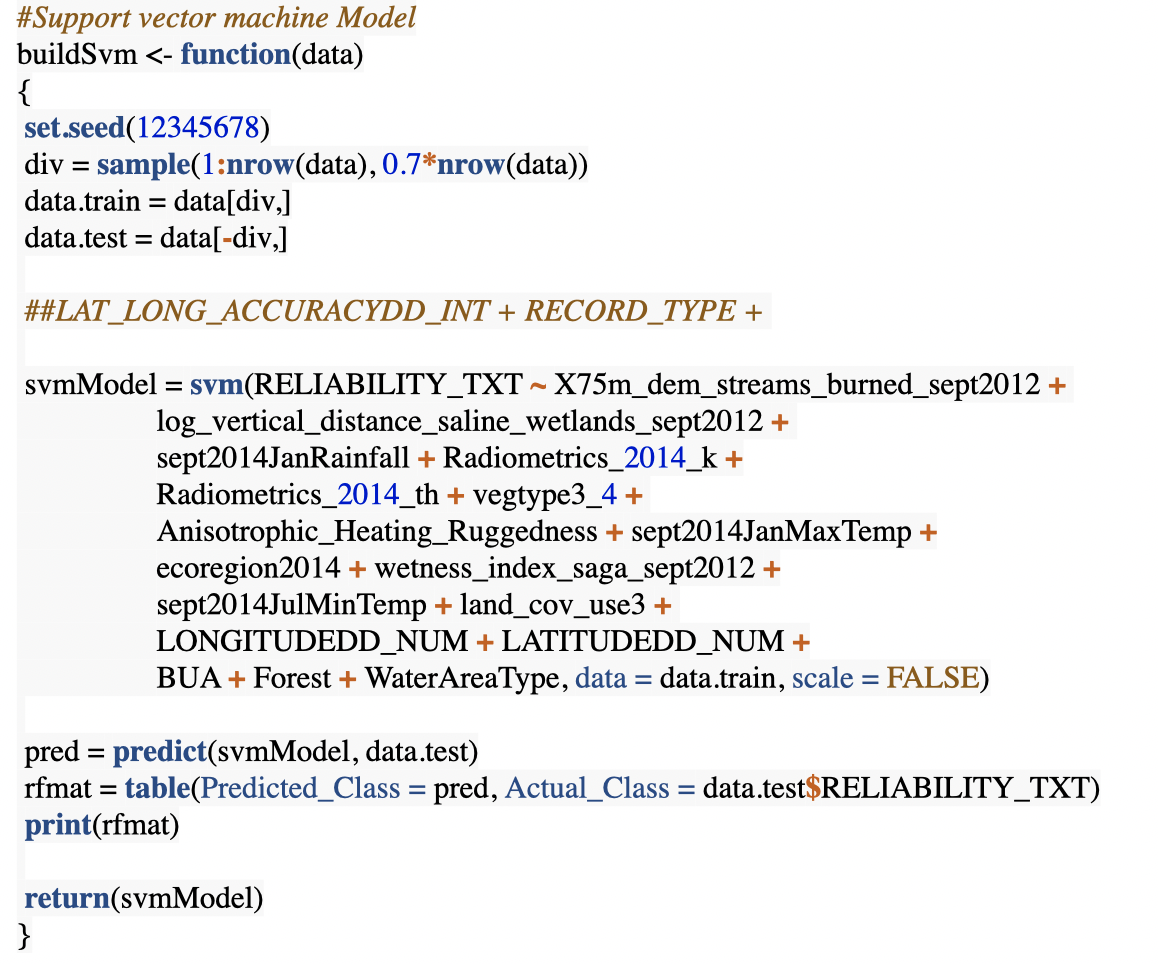
**Result:** no errors every package has been loaded. Rstudio will provide has provided some information on which function was masked from the previous libraries. This is no problem to use since we don’t use those masked functions

**Creating functions:**

**buildForest function**

****

**buildSvm function:**

****

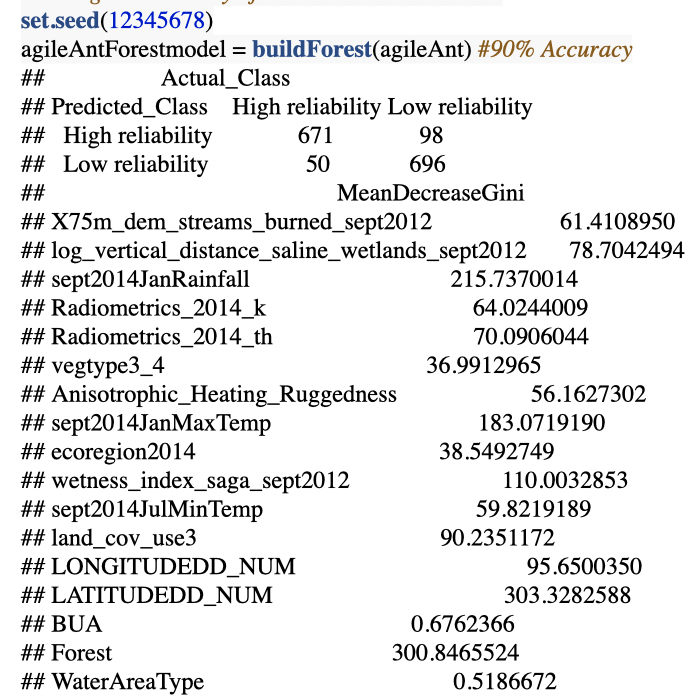
**buildXgboost function:**

**Partial code:**

****

**Testing with the species we have splitted.**

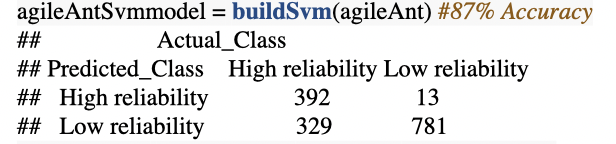
**Testing the buildForest function and it works.**

****

Result indicates the perfect execution of the function and has returned a Confusion matrix and a model importance table.

**Testing the buildSvm function:**

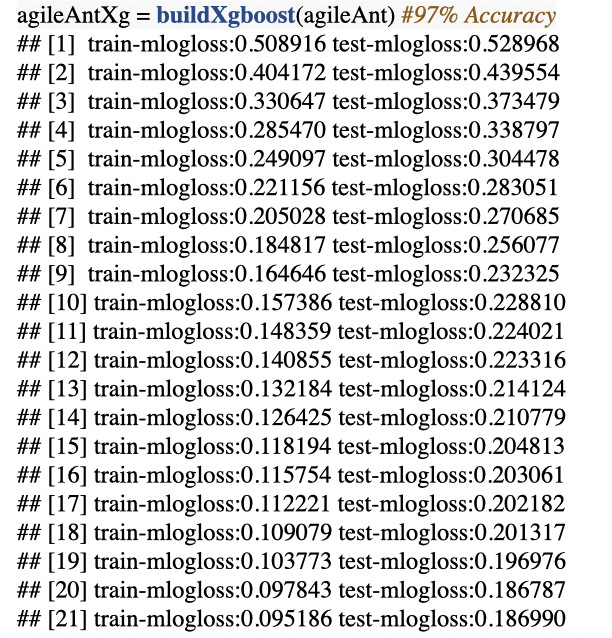
**It is expected that a confusion matrix will be returned**

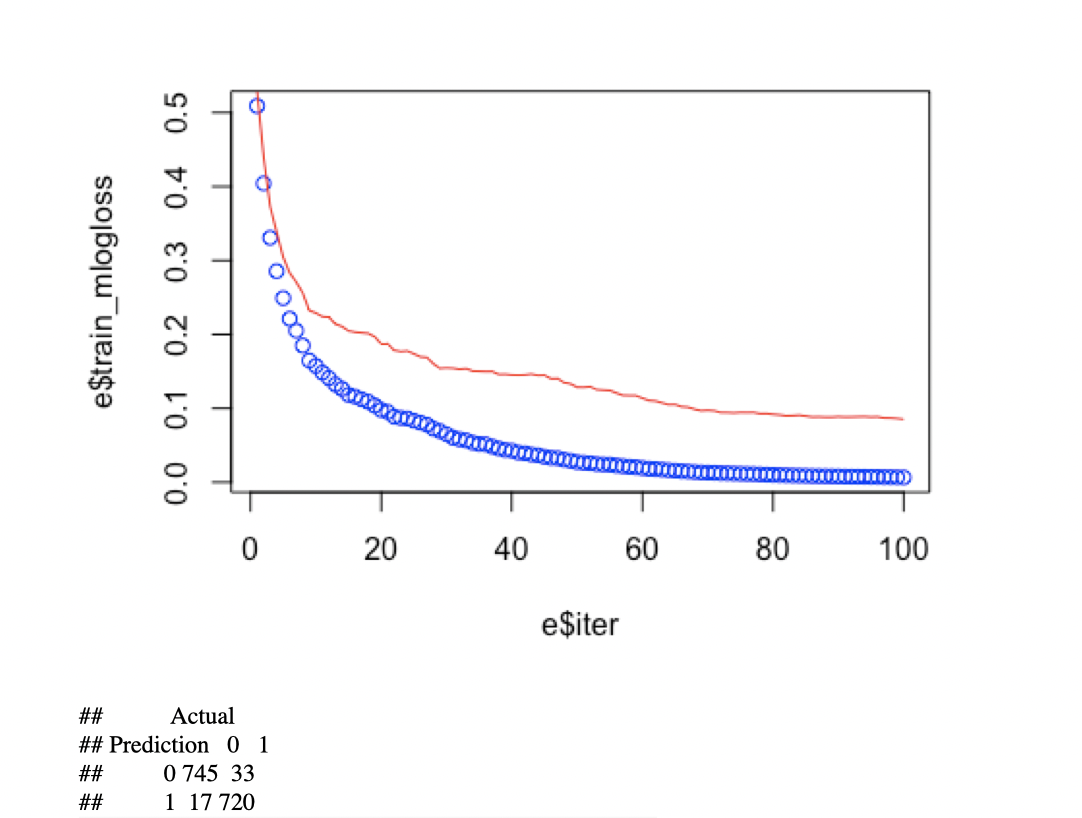
****

Result: A confusion matrix has been returned

**Testing the buildXGBoost function:**

**Expected result:** the model should show the progress and plot a plot that will show the logloss on train and test set

****

****

**Result:** The logloss at each iter is shown and drawn as well.

A confusion matrix has also been return which is expected.

Perfect execution.

These functions are then implemented into the R-Shiny which is just the user interface.

# Shiny UI Testing

User interface Test:

Chunk of code to be tested:

These are very simple page layout that was copied and modified from online sources.

These code just tells the page how to layout each button, output content window, where they are and such.

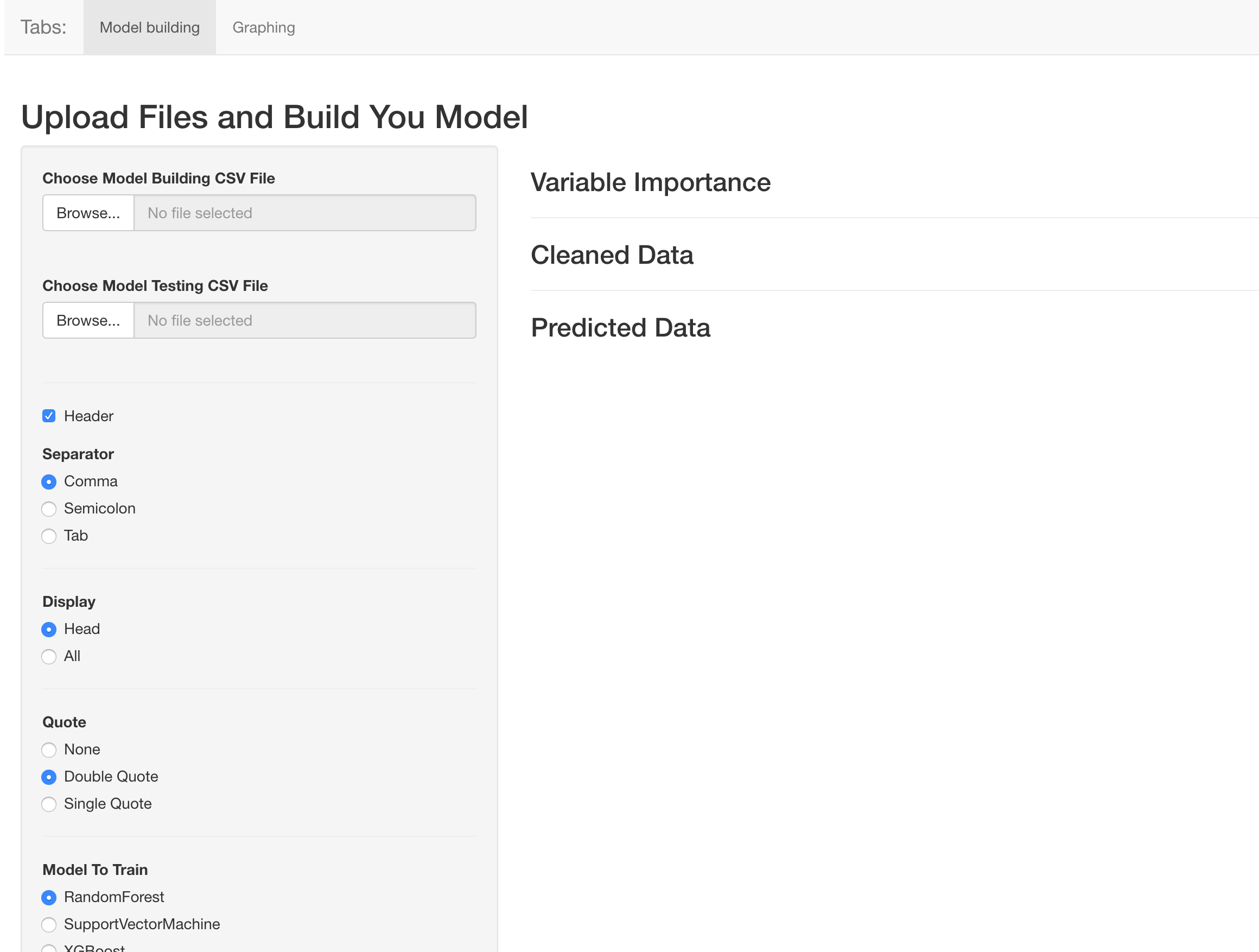


45



**Result:**

The result was expected, the layout is exactly the same as the one we had planned.

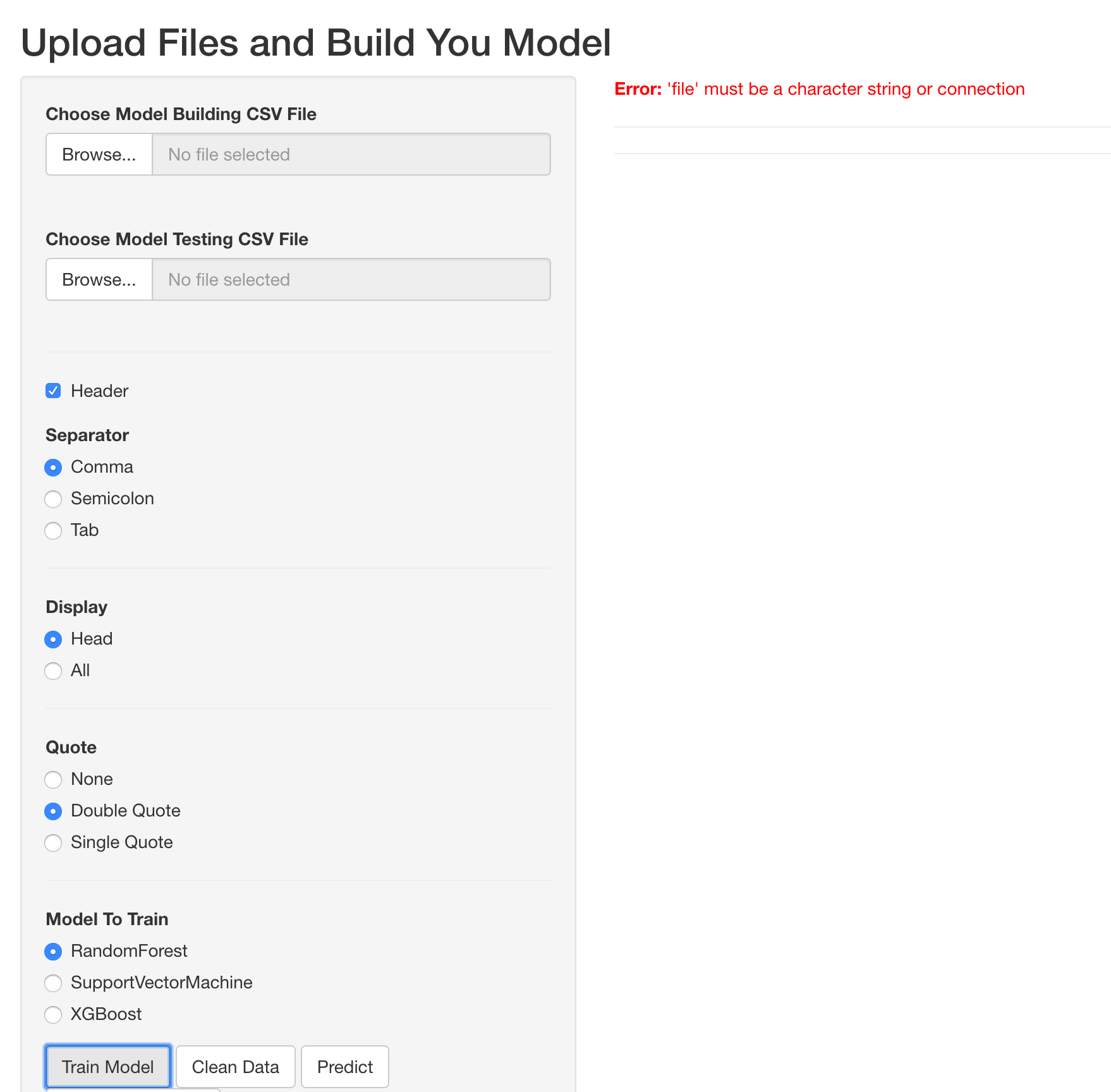


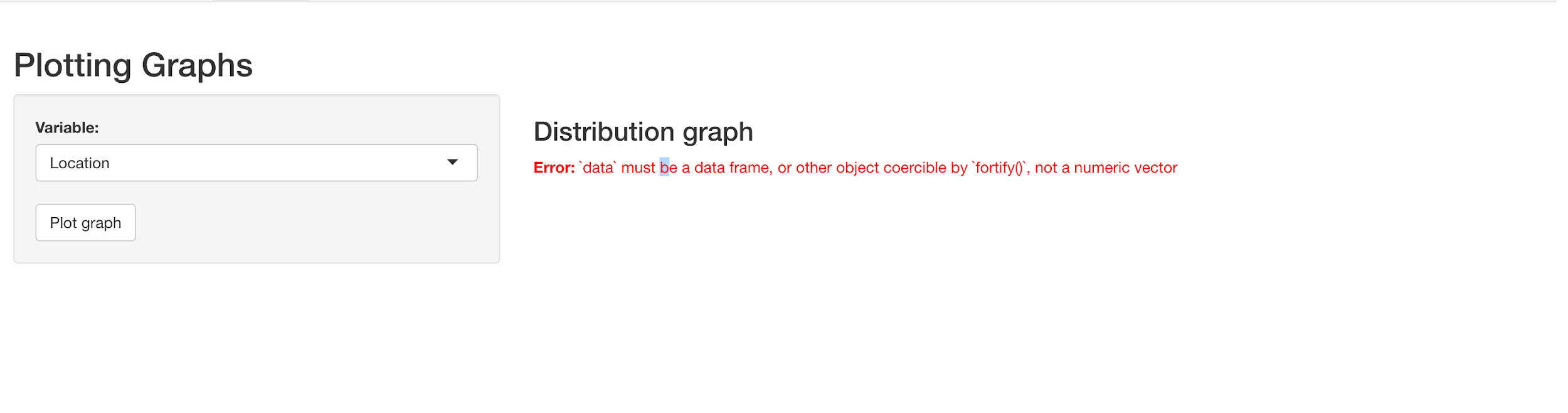
# Shiny Error Testing

When no files are provided a user can not use the Train Model, Clean Data and Predict and Plot buttons.

The system is built in a way that even if there are no file provided, the system will not crash.

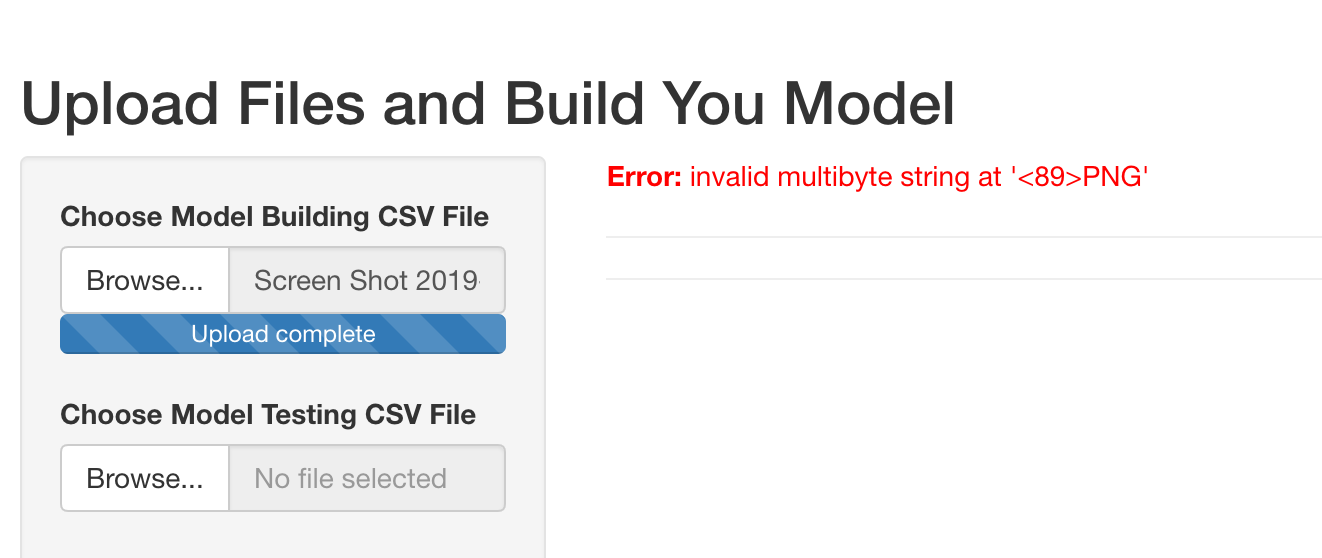
It will show an error on the user interface saying that a connection is needed. Which the user can then change and upload a file.



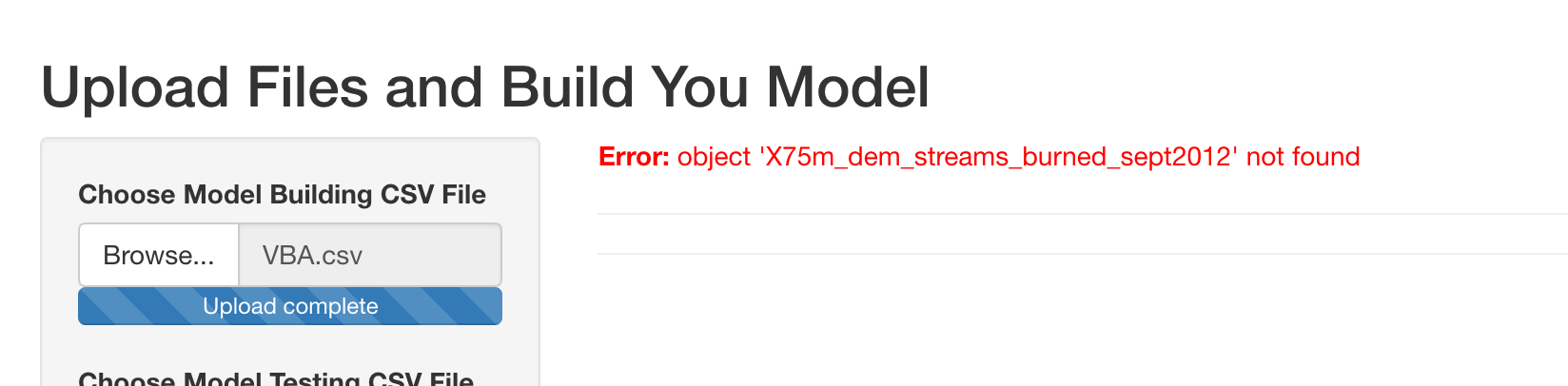


When the wrong type of files are inputted, an error will also show on the user interface.

The system is built so that it only works with CSV files.

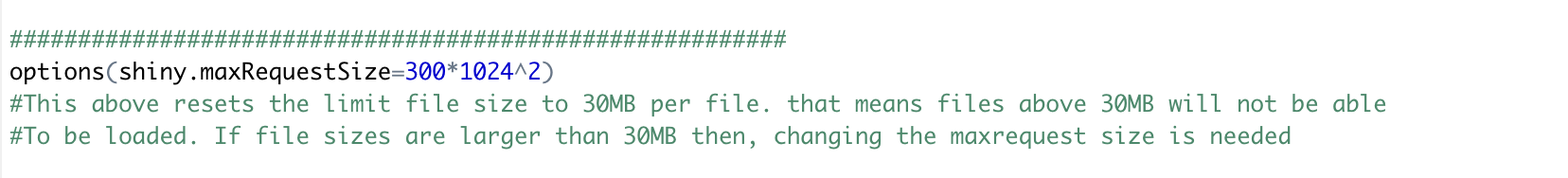


But when the input is csv files, since we have a specific formula, when the file does not have the specific columns that the model training need, it will also return an error saying that it could not find the corresponding column and it will not work.



The program is also built with a restriction of 30MB per file as the maximum file size.

A 30MB csv file would contain a lot of data and that would make our system run very slow, therefore we have at the moment capped the file size at 30MB. However, the restriction can be increased simply by changing the maxRequestSize.



Wrong input:  
When the CSV file given and the user has pressed the Clean button but the file does not have the longitude and latitude column, it will return the following error: saying that the column is not found and therefore the user must submit a new file.



**Platform Dependencies:**

Currently, the software(User Interface) only works on MacBook, it does not work on windows. There are a few bugs that we have noticed on the windows operating system which prevents the software from working properly.



The error is from function Clean, where the test data are passed to the Clean function to clean and add raster data. It appears that on the windows operating system, the raster loading method does not work properly. Therefore the system will only be able to run on MacOS.

The same error can also be reproduced after the Clean button has been pressed and the the directory given by the user does not contain raster files.

**Improvement**

Some notable improvement would be that, when we test stuff, we should print a string instead of waiting for the console output as sometimes the console does not output anything and thus the only other way to check the successful code run is from the environment tab where we can see that the data has been stored. For those functions that does print a string when it runs properly, it was easier to test those than testing just a block of code that are not included in a function.

A solution is to make a function that takes in a function and it will test the function inside and it will output a message that should tell the user whether or not the function being tested ran successfully. That would make unit testing much easier.

**Limitation**

When doing the testing we have just noticed that the maximum size for file input are 30MB which we had forgotten, so any csv file that are larger than 30MB will not be accepted.

We have also noticed that our model does not predict species that are other than the ones that we are currently given by DELWP, if the user wants to predict a new species, then the user would have to go to the data preparation Rscript(Combined.R) and add the species by themselves and then also go back to the server.R script and update the Clean function there as well as the Model building functions. This is a huge set back as the user might have a hard time navigating through our codes, but a decent amount of commenting has been provided so it should make it a bit better.

**Pros/Cons:**

The system does should the variable importances for the models except for the SVM model because the model does not focus on the variable importance function was not built in. It requires extra library to import the function that can return the variable importance however, the extra library required would cause trouble with the current SVM package.

The system does predict with models built and the user can download the predicted data to their computer if they wanted to.