Mammal Predictor

Ryan J. Cooper

11/1/2020

Contents

Introduction

This project was developed for the HarvardX Data Science Professional Certificate program, Machine Learning Capstone Project taught by Rafael Irizarry, Ph.D. and offered through the EDX online learning platform. The skills demonstrated in this report are based on lessons in this course and the accompanying text, an Introduction to Data Science. (Irizarry, 2020)

Data available from the Ecological Society of America (ESA) PanTHERIA dataset (Jones, 2016) contains descriptive attributes concerning over 5,400 individual mammal species. Additional data derived from The Global Biodiversity Information Facility (GBIF, 2020) contains extended attributes and details that will support exploration of the Pantheria dataset. These data sources will be used in the construction and training of a machine learning classification model which will try to correctly classify mammals into their most likely taxonomy given a set of identifying physical, behavioral, reproductive and life history characteristics.

- Physical characteristics include body mass, head + body length, and forearm length.
- Behavioral characteristics include population density, diet breadth and trophic level, geographic locale, and activity cycle.
- Reproductive and life history characteristics include longevity, gestation, and weaning ages.

The data set also contains data related to many other characteristics, but many of the columns do not have very complete data. The combination of these various predictors should be very informative about an animals taxonomy. Greater numbers of available predictors to train on generally should increase accuracy of predictions. However, with over 50 possible original columns available, it is necessary to identify and utilize the predictors that are most relevant, and select predictors for which there is a substantial amount of data available.

To demonstrate the concept of a classification and regression tree (CART), we will first implement the RPart package - which enable the visualization of a single decision tree using conditional probability to select the relevant variables and cut off points for each split of the tree.

After demonstration of RPART package for creation of a single decision tree, we will shift to a random forests (RF) approach that will go beyond the CART concept by combining many trees, enabling individual trees to work together, providing greater overall accuracy, but with some loss of transparency. Random forests can be surprisingly accurate in multiple outcome classification problems.

Primer on Taxonomy

Biological taxonomy is the science of classification of living organisms by related characteristics. Every organism belongs to a tree of taxa starting with the *domain* and *kingdom* at the top, with each successive (lower) level being more specific/descriptive, and generally having fewer species in it. The most specific taxonomic level in general use is *species*.

The *species* is expressed as a binomial (two word) title which incorporates the *genus* into the title. The *species* name will formally have the first word capitalized and the second word in lower case, which is the word that differentiates this species from others in the same *genus*.

For example, a blue whale is of the *genus:* Balaenoptera, *species: Balaenoptera musculus*. The last word of the species name is also referred to as the *specific epithet*. In text, the species is traditionally italicized. However, for the purpose of this study, species names will not always be italicized. In some references the *genus* part of the binomial may be shortened to a single initial, for example, *Balaenoptera musculus* may be shortened to *B. musculus*.

The 8 main levels or *Ranks* of taxonomy are: Domain > Kingdom > Phylum > Class > Order > Family > Genus > Species

Example Taxonomy of a Blue Whale: Eukarya > Animalia > Chordata > Mammalia > Cetacea > Balaenopteridea > Balaenoptera > musculus

All data in the ESA dataset is concerning animals from the *class* Mammalia (mammals). The focus of this project will be to generally identify the remaining taxonomic ranks: *order, family, or genus* given a limited number of variables available for prediction.

Please note, since the advent of widespread genetic testing, some re-organization of taxa has been occurring to better represent *monophyletic groups* or *clades* which share a common ancestor. One example of this is the order *Soricomorpha*, which appears in the Pantheria dataset, but has been partially reclassified since. The Pantheria dataset was not selected for relevance and currency - but because its hierarchical structure, sparseness, and imbalanced classes makes it an interesting subject for demonstration of random forest models.

Strengths of CART & Random Forests in Taxonomy

If the purpose of taxonomy is to differentiate species by combinations of unique markers, then a CART / RF approach should work well. Some individual characteristics are very predictive, when classes are very distinct such as the Cetacea; for example, a mammal with a body mass of greater than 100,000,000 grams is *always* going to be a blue whale. A mammal with a body mass of 20,000,000 grams, could be one of several types of whales - bow heads, right whales, etc. So this body mass must be combined with another measurment like gestational age to be a useful predictor. Many characteristics may not be very predictive, unless they are used in conjunction with other characteristics. A CART / Random Forest approach should reveal which variables are most important, and provide good prediction accuracy by finding the combinations of observations that most efficiently answer the question - which order, family and genus does this mammal belong to?

Limits & Challenges

This model is not intended as a species-level classification tool, as it contains only one row with median values for any given species. A reasonable population of data is needed for any outcome class.

The data is incomplete and any analysis performed may suffer from the *"curse of dimensionality"* - of the characteristics selected, many observations are present on only a portion of the rows. As the number of columns increases, the data becomes relatively sparse.

The number of species per taxonomical order varies greatly, with some orders having only one or two sub-families, genus, or species, and some having many individual species. This results in *unbalanced classes*, which may impact the accuracy of the final model. Predicting Chiroptera (bats) or Rodentia (rats, mice) has a much greater chance of being correct vs. guessing an obscure order with only a few species.

Model Goals

Since there are many possible outcomes, it is necessary to frame the problem carefully: What are we trying to accomplish - what are the goals of the model?

- Accuracy: A measure of overall accuracy is one of the goals - we would like a model that predicts the correct order or the correct family most of the time.
- Diversity of Outcomes: We would like a model that will predict a diverse number of orders. Given the presence of many imbalanced classes, we could make a model that predicts Rodentia or Carnivora every time, and it would be correct much of the time. But what good is a model that looks at an elephant and predicts it is a rat or a wolf? We must balance the need to include minority classes. Increasing the selection of minority classes can decrease accuracy - So accuracy and diversity of outcomes are two competing forces in this model.
- Balance: We would like a model that balances precision and recall. The F1 score is a good measure of this. We will record the average of the balanced F1 score for predicted class. If 8 classes are predicted there will be 8 F1 scores. The mean of those scores will be recorded and considered during model analysis.
- Recursion: Given that there are so many families - the number of possible *family* or *genus* outcomes is too great without segmentation or recursion. We can determine the most likely order, then filter the families and re-run the model using a more focused set of data. This could be done again to predict at the genus level as well.
- Flexibility: We would like a classification model that can be flexible and take a range of inputs with as little or as much data as is available. This may necessitate a more complex approach to modeling that can adapt to a variable number of inputs, and handle missing data effectively.

Data Setup

In this section we will load data, libraries, and packages, and set up the data for analysis.

Load Packages

Several packages and external libraries are used by this project. Key utility packages include GGPlot2 for visualizations, dplyr for data manipulation, and the caret package for various functions. The randomforest and rpart packages contain key features used in the model construction and testing process.

Load & Process Mammal Data

In this section we will preprocess the data from the ESA Pantheria dataset. The data contains over 50 different variables. For the purposes of this study, we're selecting a subset of variables where observations are present for most rows. The variables selected will include the taxonomic data and numerous predictors including physical, behavioral, reproductive, and life history characteristics. The columns with enough observations will be copied into a new data frame and renamed for easier readability, and the data types and values will be cleaned up to work correctly with R.

There are 5416 total original rows of data in the Pantheria file.

Vernacular Name Lookup Function for Species Binomials

Here we set up a lookup function to find the vernacular / common name for a species binomial.

lookup
The common name for Balaenoptera musculus is Blue Whale
The common name for Rattus rattus is Black Rat
The common name for Canis lupus is Gray Wolf

This function has returned the correct common names of the various species binomials we have passed in.

Vernacular Name Index for Orders

We will also set up an index of common names for the taxonomical orders in the Mammalia family. Again, this is done to make the exploration of the data easier to understand.

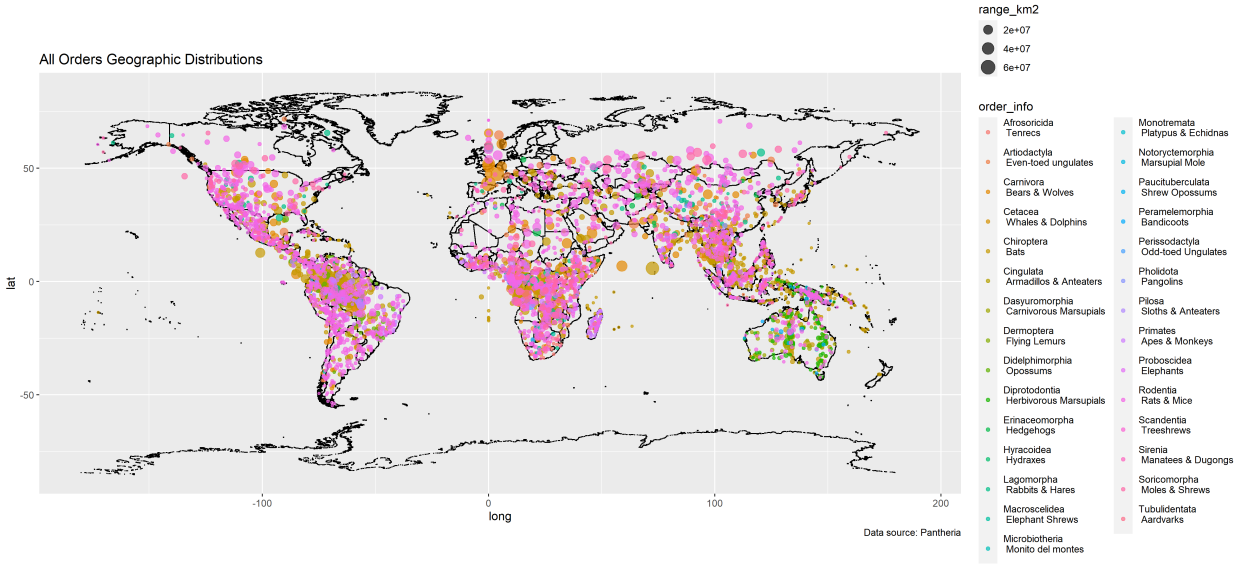
taxo_order	taxo_order_desc
Afrosoricida	Tenrecs
Artiodactyla	Even-toed ungulates
Carnivora	Bears & Wolves
Cetacea	Whales & Dolphins
Chiroptera	Bats
Cingulata	Armadillos & Anteaters
Dasyuromorphia	Carnivorous Marsupials
Dermoptera	Flying Lemurs
Didelphimorphia	Opossums
Diprotodontia	Herbivorous Marsupials

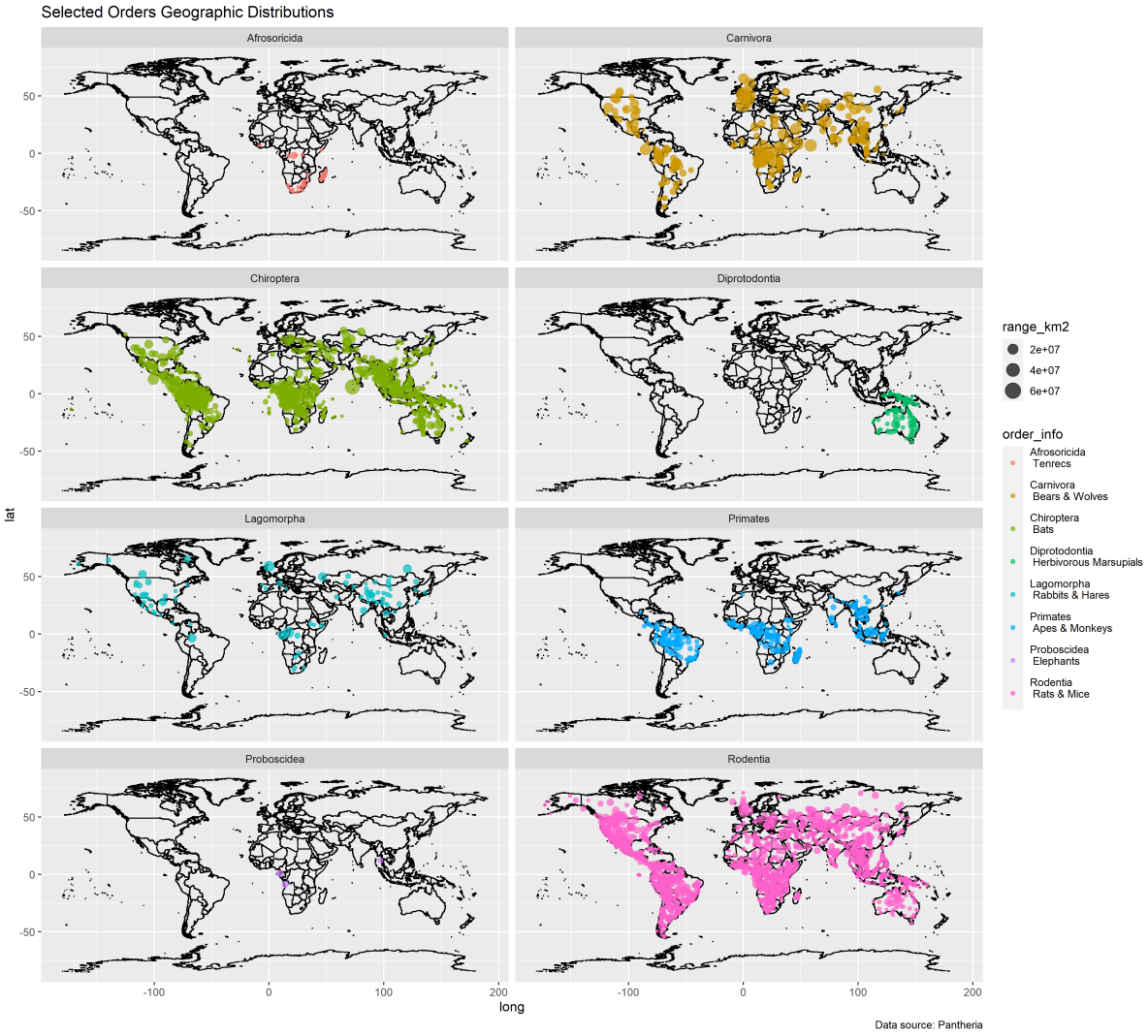
Data Wrangling & Feature Selection

In this section, we are mutating the data to be more friendly to work with. We will drop some of the fields that are not going to be used in this analysis project, and combine some of the fields into new fields which will contain a complete heirarchy of classes. Some of the data in the Pantheria datase was estimated / extrapolated - these fields are marked by the EXT. These will be coalesced with the actual values to provide the greatest final number of measurements.

Analysis

In this section we will analyze the Pantheria data to thoroughly understand the distribution, completeness, and other properties of the data.





This map shows the vast scale of the Pantheria database - it contains data related to thousands of species from all over the world. The dataset contains a latitude/longitude (geocode) for the center point of each species' known range, which have been plotted on these maps.

Some orders like Diprotodontia (Marsupials) are found only in Australia - Dermoptera (Flying Lemurs) in Indonesia & the Phillipines, Proboscidea (Elephants) in Africa and India & Tubulidentata (Aardvarks) which occur only in Sub Saharan africa.

On the other hand, some species like Chiroptera (Bats) Carnivora (Meat Eaters), Rodentia (Rats & Mice), and Lagomorpha (Rabbits) are found all over the globe. Cetacea (Whales) and Sirenia (Manatees & Dugongs) have no geocodes or range data recorded, presumably because they are marine mammals.

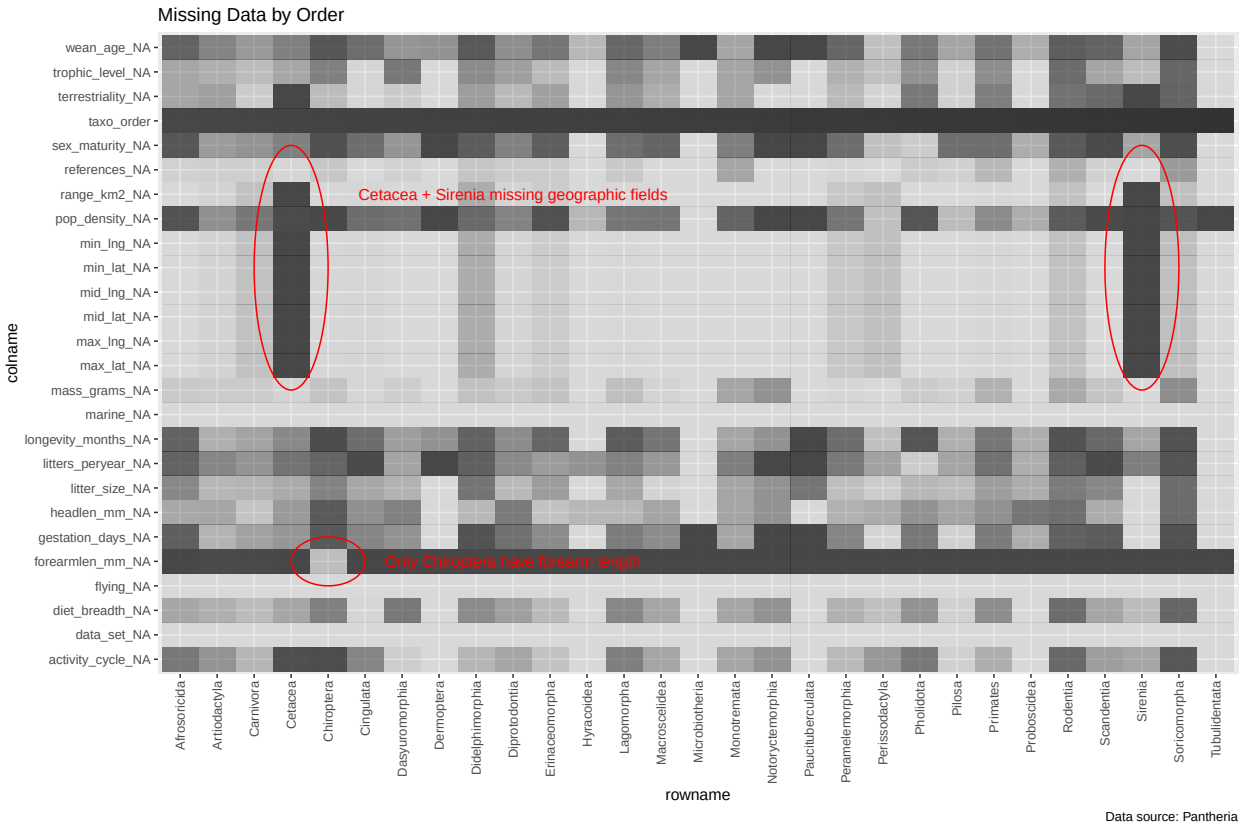
Examining Data for Completeness

Now that we've created a consolidated data set, we will analyze the details, examine the data for completeness, and try to determine what machine learning methods would be most appropriate to produce the most accurate predictions. The data used in this project comes from a scientific database which was derived from many sources.

The original source of each data reference may be found by accessing the metadata link in the references section and finding the corresponding numeric codes. The values provided by Pantheria are already highly

reduced and, in some cases, the values are the product of a regression model. The machine learning model we are building already has a lot of the data collection work "baked in" and we have just one row per species.

headlen_mm_missing	3475
forearmlen_mm_missing	4513
mass_grams_missing	1480
litter_size_missing	2915
litters_peryear_missing	4128
gestation_days_missing	4054
longevity_months_missing	4403
diet_breadth_missing	3255
pop_density_missing	4460
trophic_level_missing	3255
terrestriality_missing	2780
sex_maturity_missing	4365
wean_age_missing	4252
range_km2_missing	748
mid_lat_missing	748
mid_lng_missing	748



The dataset contains 5416 total rows with columns containing the following data points. The taxonomy follows the convention set in Wilson & Reeder's Mammal Species of the World. A Taxonomic and Geographic Reference (3rd ed), published 2005.

Class data:

- taxo_order - the taxonomic order
- taxo_family - the taxonomic family
- taxo_genus - the taxonomic genus
- taxo_species - the taxonomic species

Commonly recorded attributes:

- headlen_mm - Head & body length in mm
- mass_grams - Mass in grams
- litter_size - Number of offspring per litter
- litters_peryear - Number of litters per pear
- gestation_days - Length of gestational period in days
- longevity_months - Lifespan in months
- diet_breadth - Number of types of food sources
- pop_density - Number of individuals per square km
- trophic_level - Level in the food chain
- sex_maturity - Days to maturity
- wean_age - Days to weaning
- activity cycle - Diurnal (active during the day) / Nocturnal (active at night) etc.

Chiroptera (Bats) only:

- forearmlen_mm - Length of forearm (wing) - a key indicator in bats

Land dwellers only have geo coordinates & terrestriality:

- terrestriality - Above ground/under ground dwelling
- min_lat - Minimum latitude
- max_lat - Maximum latitude
- min_lng - Minimum longitude
- max_lng - Maximum longitude
- mid_lat - Mid range latitude
- mid_lng - Mid range longitude
- range_km2 - Range in square km

The column with the fewest missing entries is body mass. Aside from the geocodes and forearm length, the column with the most missing entries is population density.

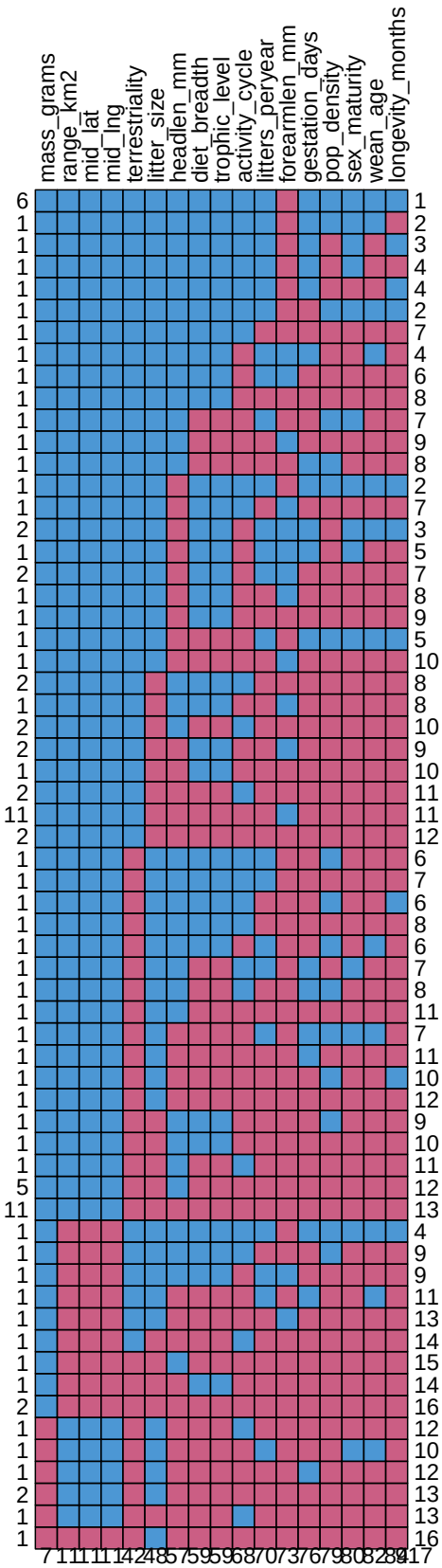
The patterns of missingness may be influenced by differences in the data collection processes for various traits. It is probably easier to ascertain certain physical values like mass (which can be measured fairly instantly) vs something like population density or longevity - which requires observation of a group, long periods of time, or complex studies to observe and record. It also makes sense that certain characteristics would not be as widely recorded for orders where it is not easily measured or not as useful, or not applicable. In particuler - flying mammals Chiroptera (Bats) and marine mammals - Cetacea (Whales) and Sirenia (Manatees & Dugongs) have some key differences in which columns are commonly recorded.

Only 269 species have all 12 common characteristics recorded. Considering the relatively small number of *complete rows* in the Pantheria data, a *complete case analysis* would be of limited use. This approach could

only incorporate a small fraction of the total observations, and the number of possible outcome classifications would include very few outcomes. This method would also require more complete data to be provided to make a prediction.

We will remove any rows that are missing all of the 12 common predictors. The original dataset contains 1278 empty rows that will be removed.

Now we will examine the pattern of missingness. We must determine if most rows are missing the same columns, and if any discernable patterns are present in the missing data.


mass_grams
range_km2
mid_lat
mid_lng
terrestriality
litter_size
headlen_mm
diet_breadth
trophic_level
activity_cycle
litters_peryear
forearmlen_mm
gestation_days
pop_density
sex_maturity
wean_age
longevity_months
6 1
1 2
1 3
1 4
1 4
1 2
1 7
1 4
1 6
1 8
1 7
1 9
1 8
1 2
1 7
2 3
1 5
2 7
1 8
1 9
1 5
1 10
2 8
1 8
2 10
2 9
1 10
2 11
11 11
2 12
1 6
1 7
1 6
1 8
1 6
1 7
1 8
1 11
1 7
1 11
1 10
1 12
1 9
1 10
1 11
5 12
11 13
1 4
1 9
1 9
1 11
1 13
1 14
1 15
1 14
2 16
1 12
1 10
1 12
2 13
1 13
1 16
7 1 1 1 1 1 4 2 4 8 5 7 5 9 5 9 6 8 7 0 7 3 7 6 7 9 8 0 8 2 8 4 1 7

Selecting 100 random rows, there are many missing values (red cells), and many different patterns of completedness in data (blue cells). The number on the left is the number of rows with that pattern of missingness. The number on the right is how many columns are missing. Missingness can be described in any of the following ways: MAR - Missing at Random, MCAR - Missing Completely at Random, or MNAR - Not Missing at Random (Rubin, 1976). For data to be considered MCAR - it must have approximately the same probability of being missing across all classes - but this is not the case here - some classes are missing greater portions of data in specific fields or groups of fields. Within these calsses, the data appears to be missing somewhat at random, so this would be MAR. Modern missing data methods usually start from the MAR assumption. (Van Buuren, 2018)

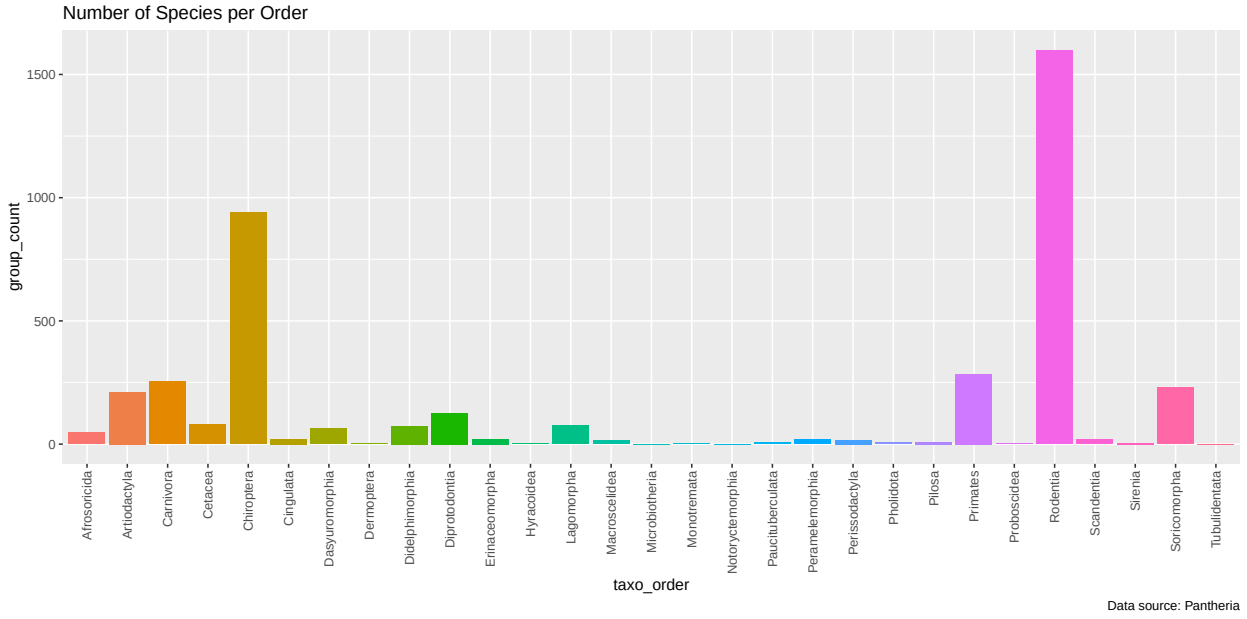
In this dataset, MAR is probably the most applicable classification, but there is also a case that some of the data is MNAR. Some variables may simply not be applicable to all species. In most classes, the missingness of rows appears to be fairly random, and is likely a natural consequence of the data collection method - this data is compiled from over 3000 reference sources. Given the fact that data did not all come from one researcher or one study - that implies that a certain amount of missingness would be expected on traits that had not been studied extensively, particularly concerning obscure or hard-to-find species. In these cases the data could also be considered MNAR as it is reflecting aspects of the data collection process - and the missingness is not at random, but a consequence of the survey methodology or other identifiable factors. For the purpose of this report, we will assume the data is MAR, but we will also try to identify and deal with data that is missing in a discernable non-random pattern.

The default approach to most regression techniques would be to omit any rows in the training data which are missing any values - This is known as *Complete Case Analysis* or may be called *List Wise Deletion*, as it is omitting entire rows from the list of training data. (Gelman, 2006) However this is not a practical approach when so few rows are complete. We will instead attempt to replace NA values with other imputed values in the training data, using various imputation strategies. We will then apply *Available Case Analysis* or *Pair-Wise Deletion* - deleting columns in the training data, to match the same columns supplied in each row of test data. This will require construction of a complex model that will pair each set of available predictor columns with only those matching data points in the training data, either real, or imputed.

Imputation of missing values could improve the number of possible outcomes, by simulating data points that were not recorded for any individual species within a given taxa. Since taxonomical classification by its very nature attempts to group similar organisms, it should be reasonable to assume that a particular species' characteristics (if missing) could be imputed based on an average of other species in the same genus, family, or order. This approach should work if the genus, family or order contains relatively similar animals - but may fail in recognizing correlations between specific observed traits, and other missing traits on the same row of data. Specifically - using an average head-body length from the genus or family, may be less accurate than say, estimating the head-body length based on a knowm, highly correlated variable such as mass. Imputation can be a complex, and subject to many combinations of methods, even within one dataset. We will experiment with how well the system performs with various imputation methods applied.

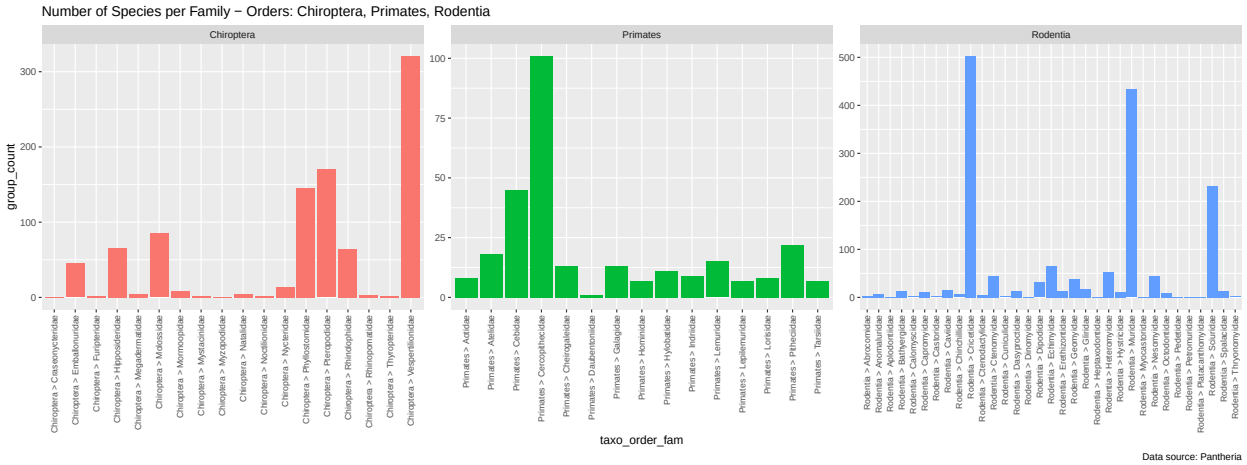
Examining Species Counts

The charts below will examine the count, distribution and ranges of values in our data. In this section we will consider the number of species per order.



The median species count is 20 species per order. We will consider any groups with fewer than the median group count, to be minority classes. This will be the basis for oversampling the training set. Of 29 orders in the data, 14 could be considered minority groups.

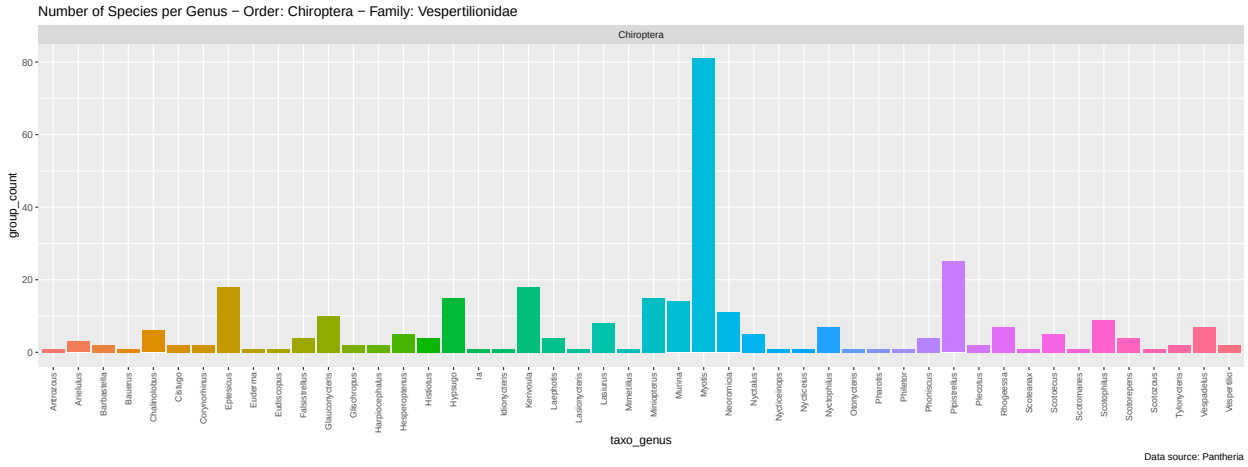
We are now performing the same analysis as above, but at the family level. Since there are so many outcomes, we will need to filter this chart to a subset of orders. We will just examine the 3 largest orders - Rodentia, Chiroptera and Primates.



Clearly the classes are very imbalanced. Some have one or two species, and others have hundreds.

The median number of families per order across all species is: 7. We may also use this detail as the basis for some imputation strategies to be explored later in this report.

We are now performing the same analysis as above, but at the genus level.



Once again we can see that there is significant imbalance in the number of genus groups per family.

Visualization of Trait Distributions

In the following plots we will examine the taxonomic data to visualize how the distributions vary between each order. First we will examine some of the raw data, to make sure the values agree with general knowledge.

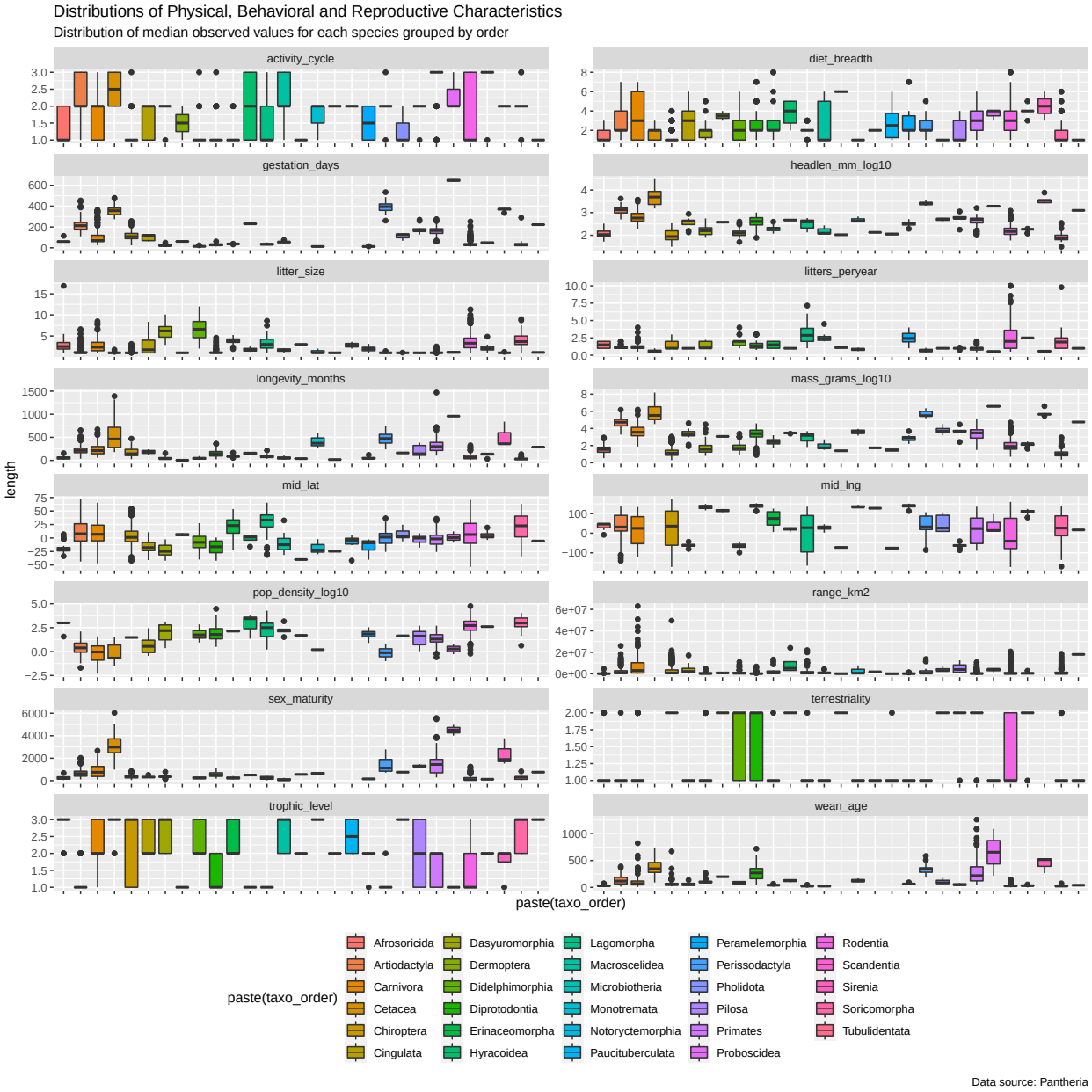
stat	maxspecies_vernacular	maxstat	minspecies_vernacular	minstat
mass_grams	Blue Whale	154321304.50	Bumblebee Bat	1.9600000
headlen_mm	Blue Whale	30480.00	Bumblebee Bat	30.9900000
forearmlen_mm	Large Flying Fox	200.00	Mouse-like Pipistrelle	23.0000000
litter_size	Tail-less Tenrec	16.89	African Bush Elephant	0.8400000
litters_peryear	Royle s Mountain Vole	10.00	Sperm Whale	0.2500000
longevity_months	Human	1470.00	Sunda Flying Lemur	3.4500000
gestation_days	African Bush Elephant	660.00	New Guinean echidna	10.0000000
wean_age	Chimpanzee	1260.81	Southern Moutain Viscacha	1.9400000
pop_density	Eurasian Water Vole	57067.85	Hooded Seal	0.0048221
trophic_level	Congo Golden Mole	3.00	Pronghorn	1.0000000
terrestriality	Lesser Hedgehog Tenrec	2.00	Hottentot Golden Mole	1.0000000
diet_breadth	Eastern hedgehog	8.00	Congo Golden Mole	1.0000000
activity_cycle	Impala	3.00	Stuhlmann s Golden Mole	1.0000000
range_km2	Red Fox	63034304.34	Bramble Cay Melomys	0.0001874
mid_lat	Musk ox	71.68	Magellanic Pygmy Rice Rat	-53.7500000

The table above examines the data range for each predictor (the minimum and maximum of each variable). Most of the details in the the table seem to agree with general knowledge. Mass and body size reveal the huge Blue whale vs. tiny, lightweight Bumblebee Bats. Examining Litter Size, Litters per year, Gestation, and Weaning age - Large mammals including Elephants and Sperm Whales, which reproduce infrequent small litters (K-selected species) appear opposite diminuitive Mice, Tenrec, and Echidnas (r-selected species) which are much more prolific in reproductive frequency. Chimpanzees are another K-selected species with a long weaning age vs. the very fast reproducing Viscacha, who wean their young for a very short period. Population density statistics indicate the Hooded Seal as the most solitary vs highly communal Water Voles.

The following box plots and scatter plots show considerable diversity in the distributions, types, and ranges of the various data points for each taxonomical *order*. There are 29 orders represented in our dataset. The box plots indicate the inter-quartile ranges of each "creature feature" including continuous-valued variables like mass & length, as well as discrete-valued variables like terrestriality, diet breadth, and activity cycle.

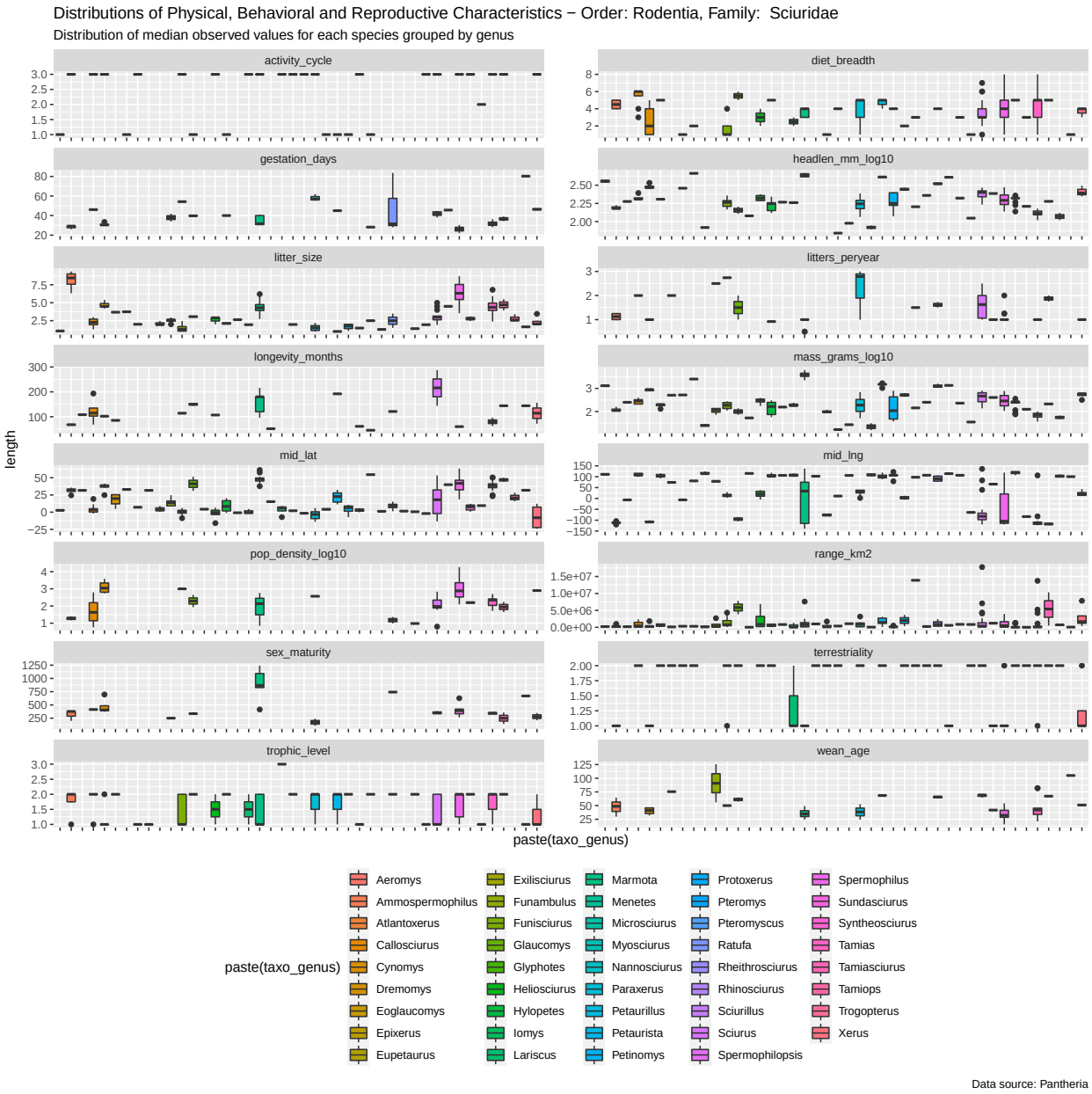
The aim of a machine learning model would be to find patterns in these features that are predictive in a way that's useful.

In general - determing the taxonomical order is not terribly difficult from a basic vislal inspection- For example - sharp canine teeth mark Carnivora. Mammals with wings are Chiroptera. Hooved animals with even number of toes are always Perissodactyla. A classification tool determining *order* may be of limited value - though it could be useful in some cases where two orders may contain similar species. For example, Scandentia, Soricomorpha, Paucituberculata, Macroscelidea all contain shrews. So we might expect to see more confusion between these similar orders.



The difference between families are much more specific, and can be used to differentiate more similar species. We will examine some characteristics of families within the largest order, Rodentia.

Predicting the family correctly could be more helpful in distinguishing fairly similar species - for example voles (Cricetidae), Rats (Muridae), and Gophers (Geomyidae) are all similar *families* in *order* Rodentia.



And finally, looking at one *family* of Rodents - we can see that the data becomes more sparse at the *genus* level, with many columns containing missing data. There also appear to be many genera that have only one value recorded for all rows (as indicated by a horizontal bar). The genus is the most specific rank that would be practical to predict in a machine learning context given this data set.

These charts suggest that a recursive model might also be practical. There are too many classes of genus to predict that right off the bat - so the first model will train on all data to predict the order, then retrain once an order has been predicted, with data that only includes the families within that one order, then again at the family level to predict the genus. Orders with more complete data should produce more accurate results when recursed.

Examining Correlations

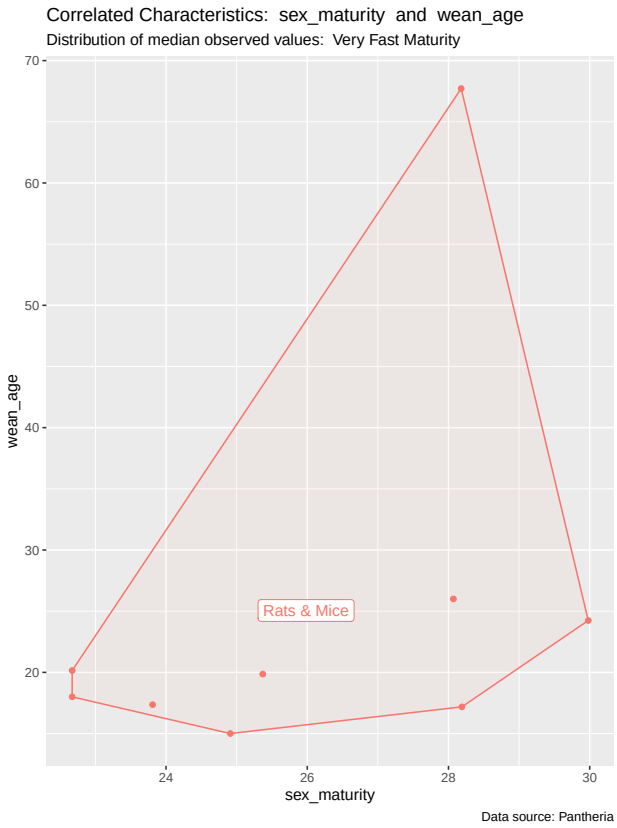
The charts below visualize some of the most highly correlated variables and how their distributions are grouped at the order level. We will consider how closely various predictors are correlated, and understand their particular attributes of covariance.

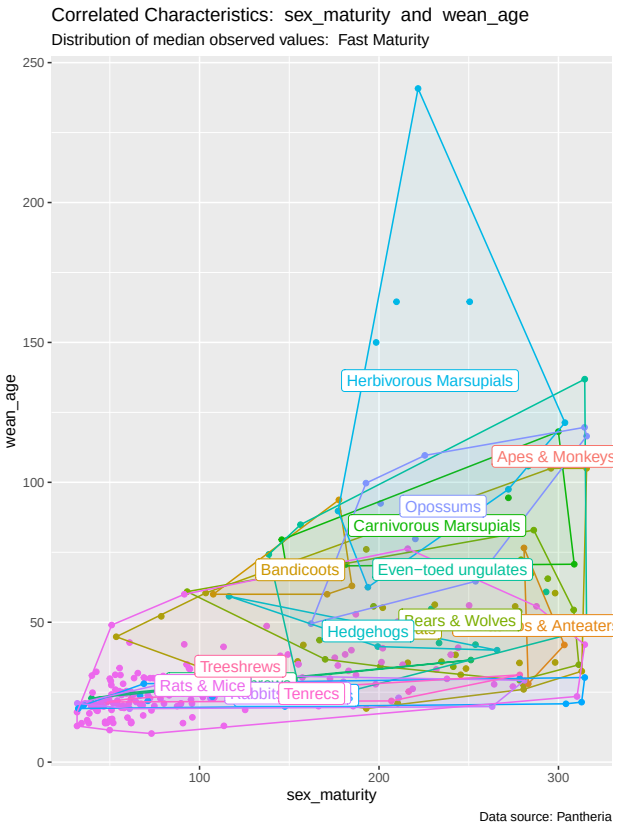
```
## # A tibble: 16 x 17
##    rowname mass_grams headlen_mm litter_size litters_peryear gestation_days
##    <chr>        <dbl>      <dbl>       <dbl>           <dbl>          <dbl>
##  1 mass_g~   NA           0.762      -0.0464         -0.0750         0.187
##  2 headle~    0.762      NA          -0.203          -0.227          0.548
##  3 litter~   -0.0464     -0.203      NA               0.251         -0.552
##  4 litter~   -0.0750     -0.227       0.251          NA             -0.422
##  5 gestat~    0.187       0.548      -0.552          -0.422         NA
##  6 wean_a~    0.0801      0.266      -0.368          -0.359          0.552
##  7 sex_ma~    0.244       0.502      -0.400          -0.449          0.719
##  8 longev~    0.367       0.670      -0.456          -0.417          0.696
##  9 pop_de~   -0.0217     -0.147       0.204           0.253         -0.185
## 10 diet_b~   -0.0348     -0.0365      0.162           0.0332        -0.0205
## 11 terres~   -0.139      -0.318      -0.326          -0.133         -0.0290
## 12 trophi~    0.0640      0.0789      0.0951         -0.159         -0.0940
## 13 activi~    0.0207      0.129      -0.0862         -0.178          0.271
## 14 range_~    0.0260      0.152       0.0230         -0.0571        -0.00330
## 15 mid_lat    0.00797    -0.0296      0.297           0.0566        -0.144
## 16 mid_lng    0.0278      0.0634     -0.0926         -0.0864         0.0722
## # ... with 11 more variables: wean_age <dbl>, sex_maturity <dbl>,
## #   longevity_months <dbl>, pop_density <dbl>, diet_breadth <dbl>,
## #   terrestriality <dbl>, trophic_level <dbl>, activity_cycle <dbl>,
## #   range_km2 <dbl>, mid_lat <dbl>, mid_lng <dbl>
```

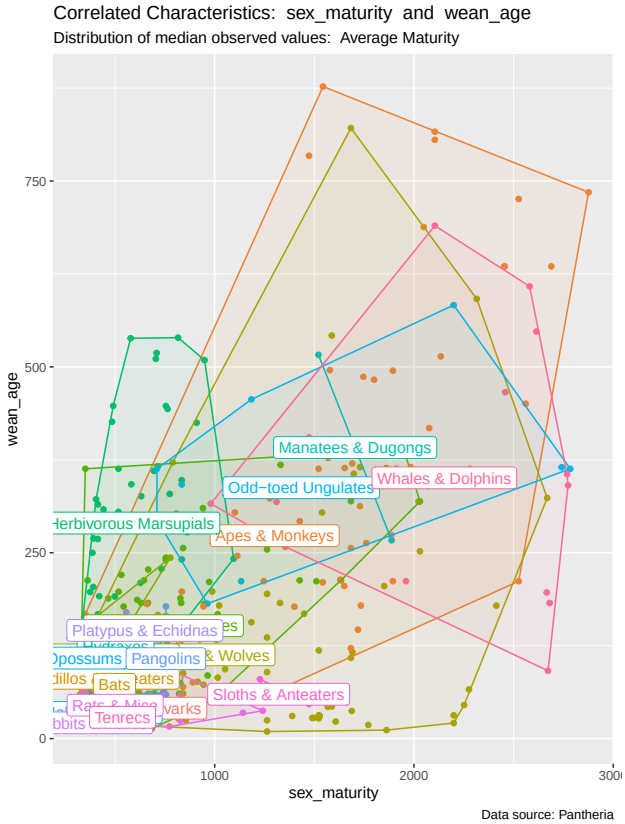
This table demonstrates that there are strong correlations between certain sets of values:

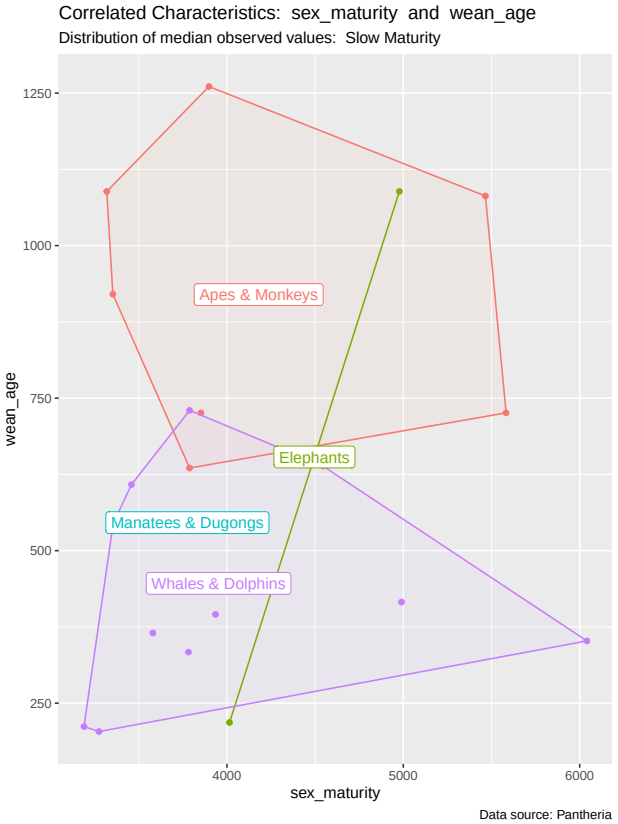
- Longevity and sex maturity: 78%
- Mass and head length: 76%
- Weaning age and sex maturity: 74%
- Sex maturity and gestation days: 72%
- Longevity and gestation days: 70%
- Longevity and head length: 67%
- Weaning age and gestation days: 55%
- Litter size and gestation days: -55%

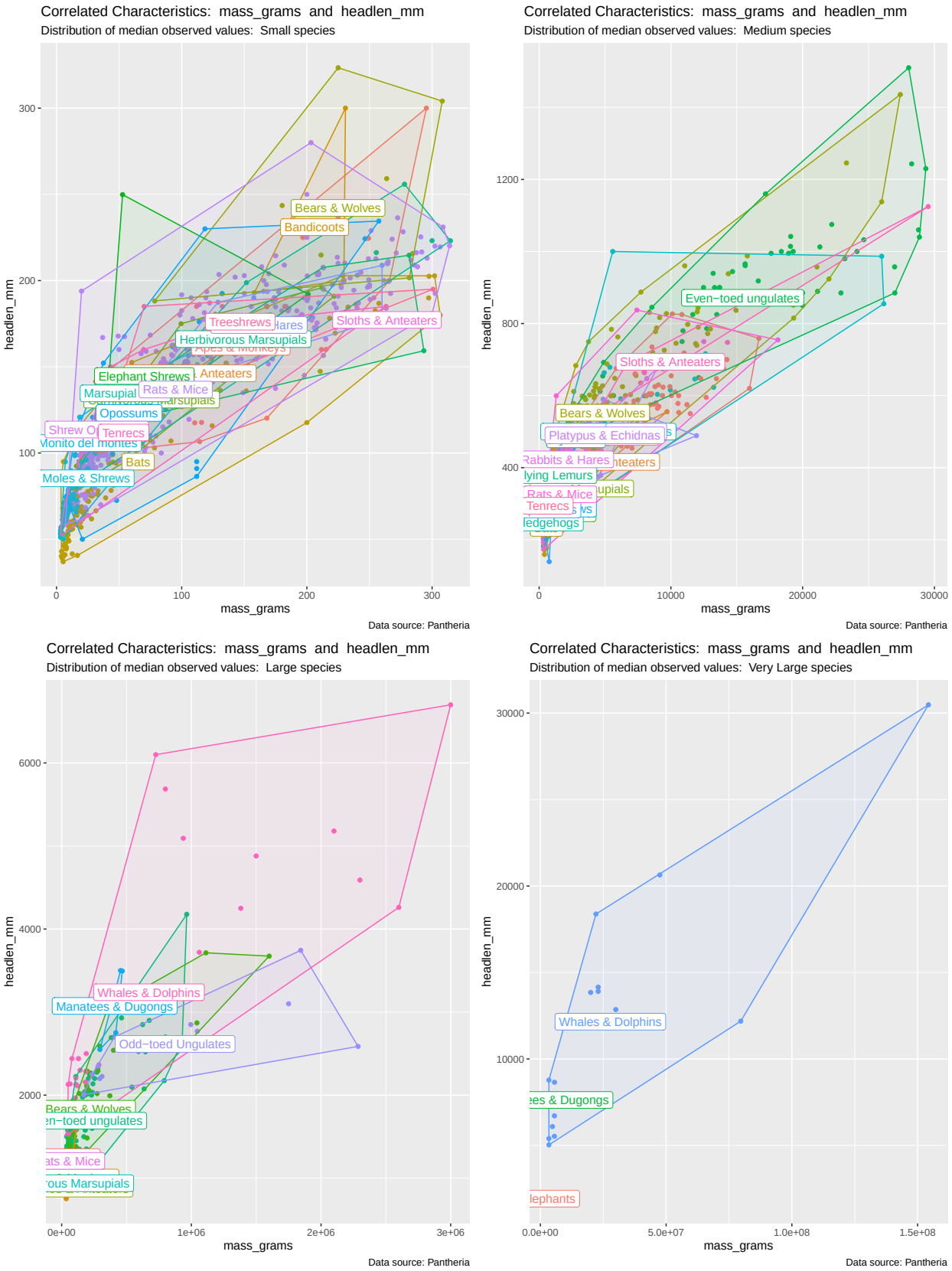
These correlations suggest that some fields may be candidates for use of linear regression to impute some missing values, since the value of one known variable may be highly correlated with teh value of an unknown one. The following scatter plots have been faceted for easier readability, but they are still somewhat crowded. The convex hull shows the outline of each species x/y distribution of two highly correlated variables.


Correlated Characteristics: sex_maturity and wean_age
Distribution of median observed values: Very Fast Maturity
wean_age
sex_maturity
Rats & Mice
Data source: Pantheria


Correlated Characteristics: sex_maturity and wean_age
Distribution of median observed values: Fast Maturity
wean_age
sex_maturity
Herbivorous Marsupials
Apes & Monkeys
Opossums
Carnivorous Marsupials
Bandicoots
Even−toed ungulates
Bears & Wolves
Hedgehogs
Treeshrews
Rats & Mice
Tenrecs
Data source: Pantheria


Correlated Characteristics: sex_maturity and wean_age
Distribution of median observed values: Average Maturity
wean_age
sex_maturity
Manatees & Dugongs
Whales & Dolphins
Odd−toed Ungulates
Apes & Monkeys
Platypus & Echidnas
Pangolins
Bats
Sloths & Anteaters
Tenrecs
Data source: Pantheria


Correlated Characteristics: sex_maturity and wean_age
Distribution of median observed values: Slow Maturity
wean_age
sex_maturity
Apes & Monkeys
Elephants
Manatees & Dugongs
Whales & Dolphins
Data source: Pantheria



Examining the most highly correlated values reveals that there are some clear patterns of correlation between

orders and families, but there is also significant overlap between the orders. Based on this analysis, a classification tree or random forest approach could be useful in finding important predictive variables to locate logical groupings and isolate areas where predictions can be made accurately, by implementing multiple predictors in a decision tree approach.

Model Construction

In this section we will begin constructing and testing models. The first approach will be to use a simple Classification and Regression Tree (CART). We will assess the accuracy values of the CART, and then examine how a random forest model may improve the accuracy. We create a final table of data which has ONLY the class/factor columns, and the various predictors. This data will contain only the variables we plan to use in the final model. This data will then be partitioned for cross validation, training, and testing.

To create a test and training data set that have the same classes represented, we will have to stratify the data by order, then select a portion to of each order for training, and the rest for testing. To maximize the number of observations and to ensure a fair number of challenges - we will start by splitting the data into test and training data for each order. The splitstackshape package enables stratified selection of random data points to create a hold-out data set that would contain a large variety of classes to challenge the machine learning model.

Assessing the Model Scores

The following key metrics relate to the predictions we are making:

- Sensitivity - True Positives - how often the model predicts the right order/family
- Specificity - True Negatives - how often the model avoids predicting this class incorrectly
- Precision: True Positives / True Positives + False Positives
- Recall: True Positives / True Positives + False Negatives
- F1 Score: 2 * ((Precision * Recall) / (Precision + Recall))

So our goal will be do accomplish several objectives - to *increase the number of classes with 1 or more predictions*, to improve the *Overall Accuracy* of the model; to achieve a good average *F1 score* for those classes which were predicted, balancing precision and recall across all predicted classes. Several functions will be set up to analyze and assess the results. For each iteration in the model construction, we will create a grid-style heatmap plot for the confusion matrix and run a function to measure accuracy and other key metrics we defined.

Create Data Partitions

We are splitting the test and training data using the stratified method so there are some samples of the same orders in both test and training data sets.

The partition has taken the original data table of 4138 original rows and reserved a hold-out of 289 to assess the final model score. The remaining 3849 rows have been further partitioned into a set of 3078 training rows and 771 test rows to use for the model construction and tuning process. There are 25 taxonomic orders present in the test data. The hold-out validation data represents unknown information - we will not use this data for any other purpose except to test the performance of the final model.

Baseline Model

The expected value of a random classification being correct in this data set would be 0.034 at the order rank, 0.0066 at the family rank and 8.6×10^{-4} at the genus rank - not a very good chance to predict the right classes with a random guess.

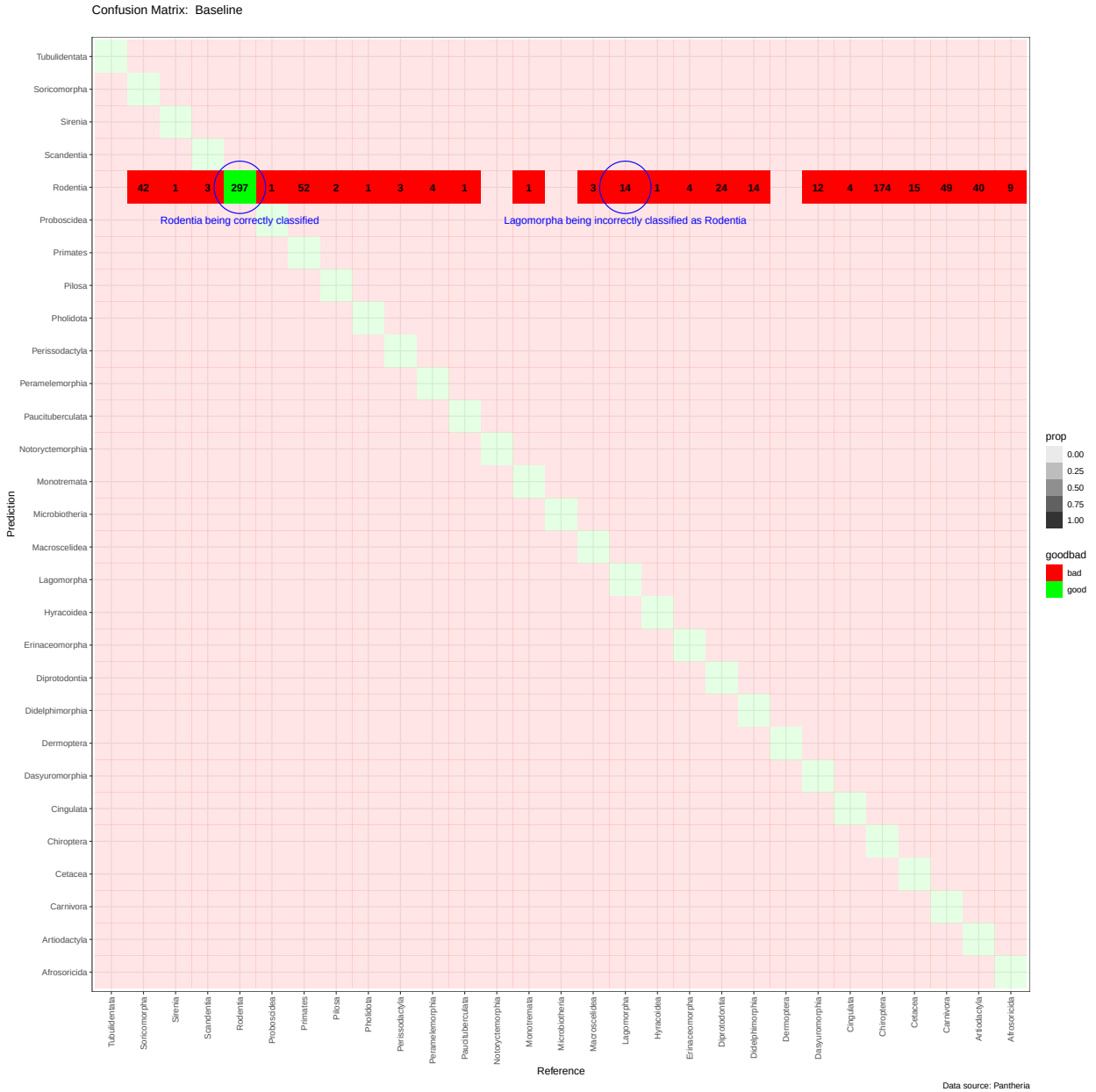
As noted previously, a model that only predicts to the order rank is not very useful, but for simplicity, we will focus only on the order rank during model construction, and then try to apply the same strategies to drill down and predict at the family and genus ranks.

To establish a baseline for performance, we will first predict the most common order every time - the following code will apply a prediction of Rodentia on every mammal (since we know they make up nearly half of all the defined species).

model	dataset	Predicted	Accuracy	F1_Mean
1- Baseline	With 0's	1	0.385214	0.5561798

Reading the Confusion Matrix Visualization Grid

The chart below visualizes the confusion matrix. A number in any cell other than a green cell, indicates a wrong guess and a number in a green cell is a correct guess. The reference or correct classes are listed on the y axis and the predictions made for each reference class are listed horizontally along the x axis. Annotations have been added on some of the confusion matrix visuals to call out specific data points or assist the reader in understanding how to read these charts.



This approach is obviously not a great choice, unless you think every Mammal is a Rat! But you can see that we would be right almost 40% of the time if you always predicted this one majority class. We will call 40% our threshold value or baseline. Now that we have established a baseline, we will build some more useful models and see if we can predict with greater than 40% accuracy at the order level.

Classification Trees

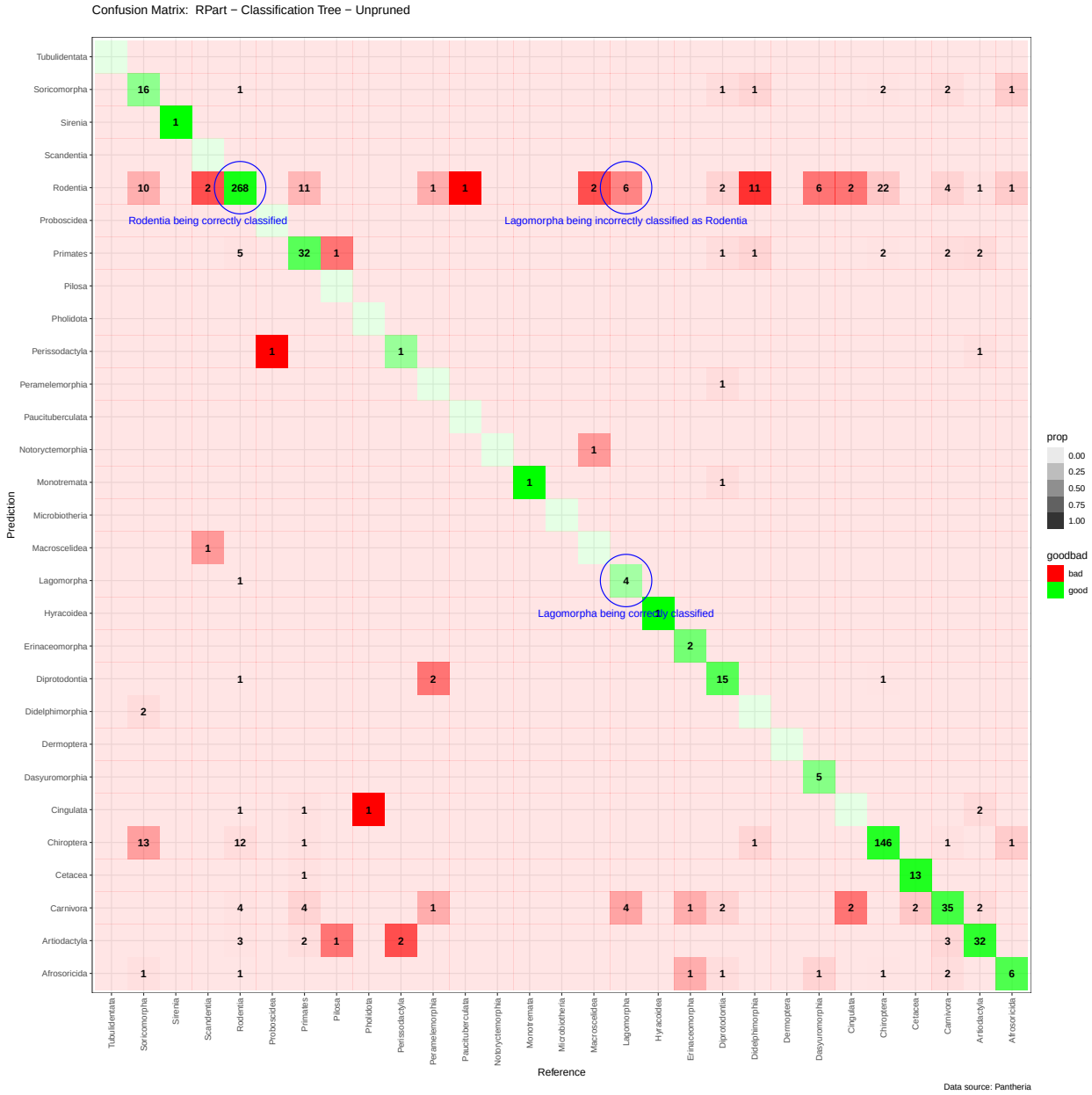
The test set contains most of the 29 orders in the training set. We'll start with a simple classification and regression tree model (CART).

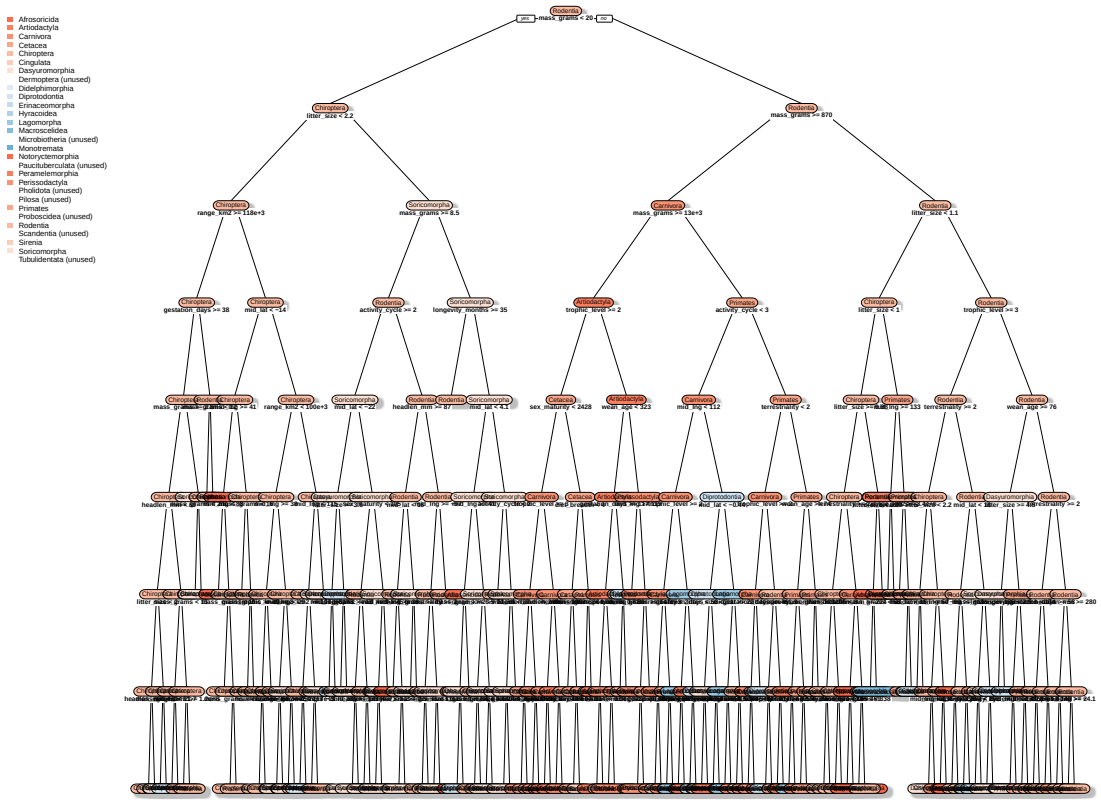
The RPart package provides features to visualize how the tree splits on each variable.

The model has been trained, now we will apply a prediction using test data. This will take the test data set, remove the order column, and get a prediction based on the remaining columns.

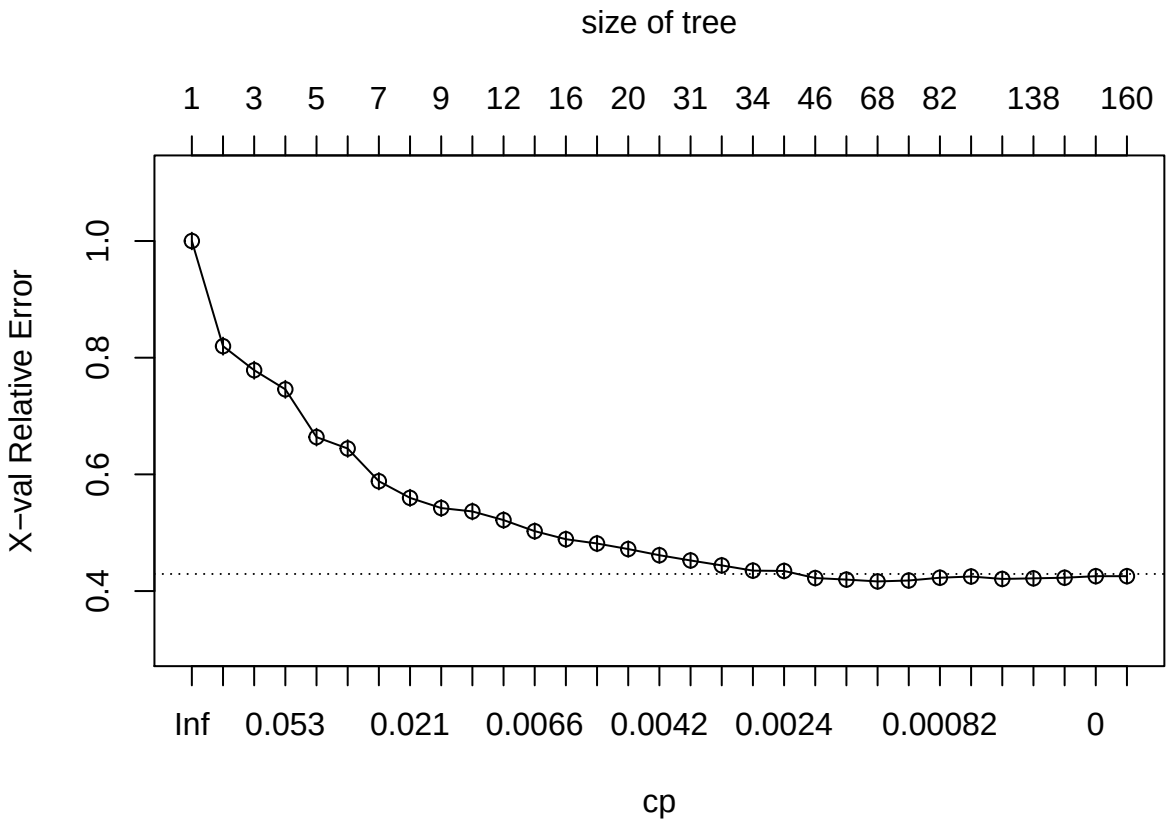
model	dataset	Predicted	Accuracy	F1_Mean
1- Baseline	With 0's	1	0.3852140	0.5561798
2- CART - Unpruned	With 0's	16	0.7496757	0.6891504

This CART results in about 70% accuracy. This is pretty good for a single decision tree - but this decision tree is overly complex and overfitted. It would need to be pruned to be more flexible.

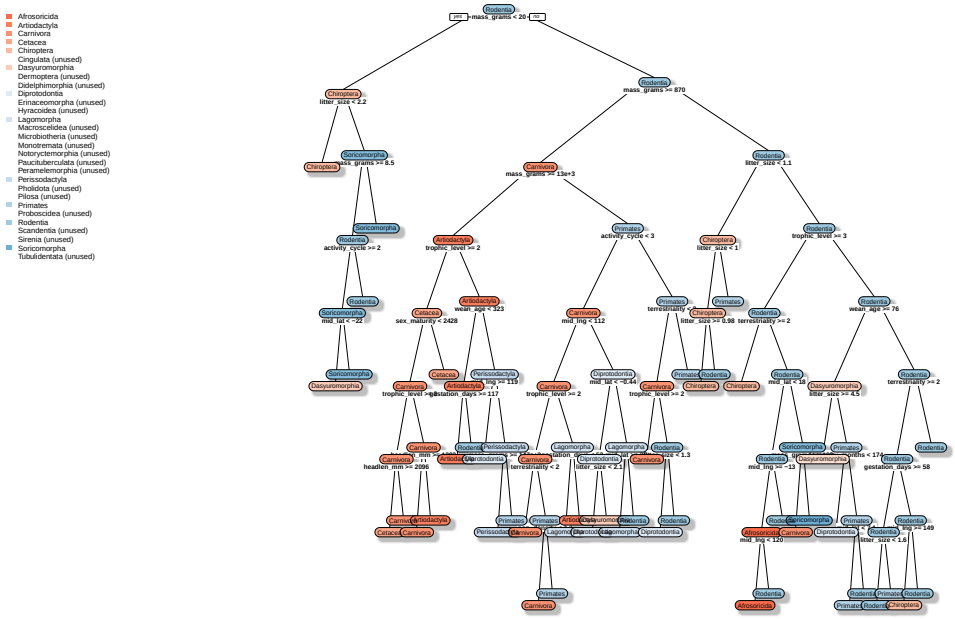

Confusion Matrix: RPart − Classification Tree − Unpruned
Prediction
Reference
Tubulidentata
Soricomorpha
Sirenia
Scandentia
Rodentia
Proboscidea
Primates
Pilosa
Pholidota
Perissodactyla
Peramelemorphia
Paucituberculata
Notoryctemorphia
Monotremata
Microbiotheria
Macroscelidea
Lagomorpha
Hyracoidea
Erinaceomorpha
Diprotodontia
Didelphimorphia
Dermoptera
Dasyuromorphia
Cingulata
Chiroptera
Cetacea
Carnivora
Artiodactyla
Afrosoricida
Rodentia being correctly classified
Lagomorpha being incorrectly classified as Rodentia
Lagomorpha being correctly classified
prop
0.00
0.25
0.50
0.75
1.00
goodbad
bad
good
Data source: Pantheria



The unpruned classification tree is very complex, and many orders are excluded. This plot demonstrates one challenge of using decision trees with sparse data over many outcomes. The classification above could be considered over-fitted. It is too complex and the number of splits is too great. It's going to match the training data very closely. Pruning the tree could reduce complexity to a more practical level to make the model more robust.



According to the RPart documentation for the plotcp feature "A good choice of cp for pruning is often the leftmost value for which the mean lies below the horizontal line." Considering the plotcp above a reasonable complexity parameter would be around .002 which is near the left most reading where the X-val Relative Error curve crosses the dotted line.

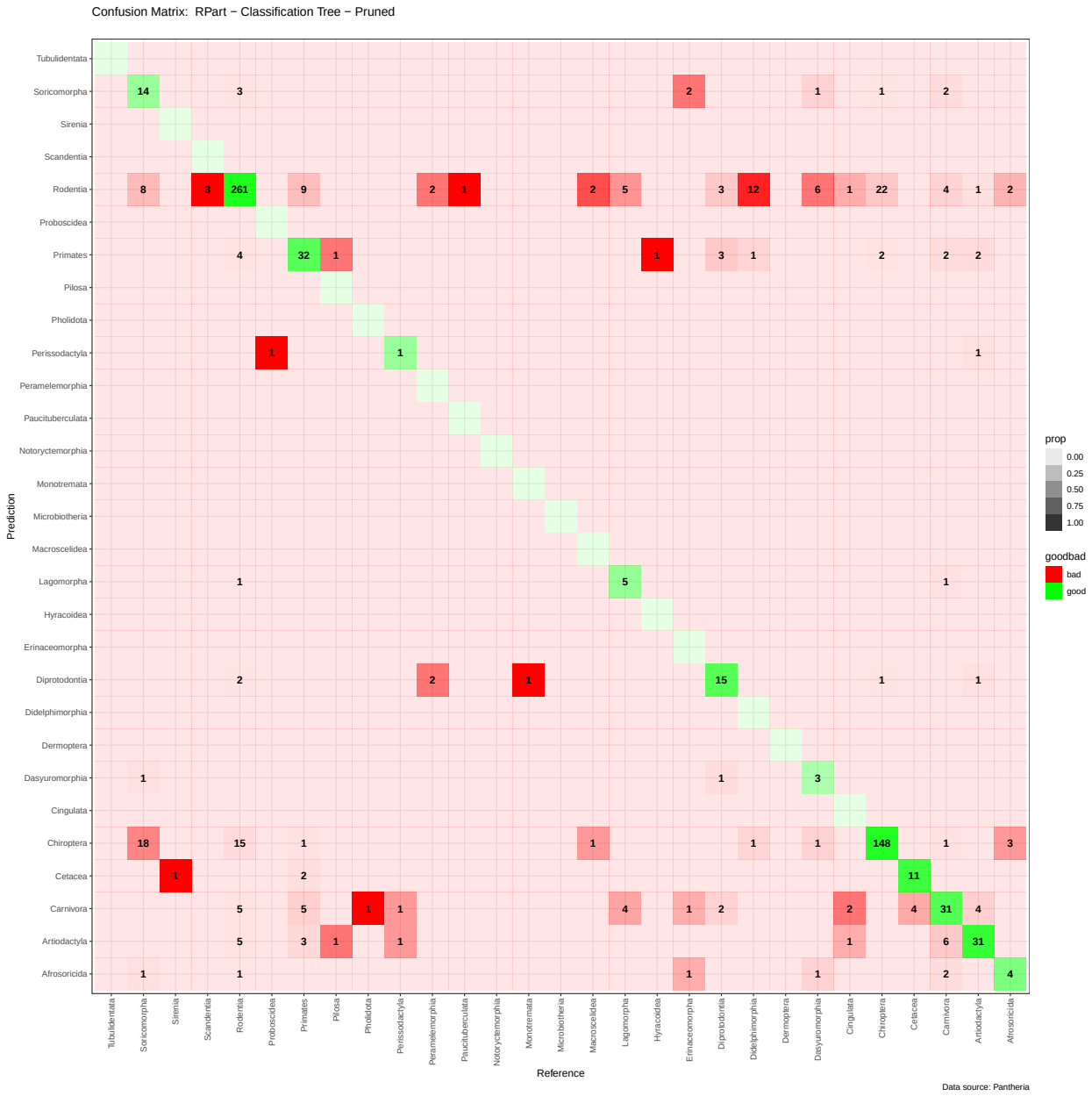


To read the tree above, you must start at the top. Like our baseline model, the model will predict Rodentia to begin. It will first look at mass grams, and if < 20, it's going to take the path to the left and change the prediction to Chiroptera. Then it will examine the litter size, if < 2.2, it's taking the left path again, still predicting Chiroptera. There are no more nodes under this one - so it's done - Chiroptera is the prediction. However, if litter size > 2.2 then it's going to follow the path on the right and change the prediction to Soricomorpha, and so on, continuing down the tree.

model	dataset	Predicted	Accuracy	F1_Mean
1- Baseline	With 0's	1	0.3852140	0.5561798
2- CART - Unpruned	With 0's	16	0.7496757	0.6891504
3- CART - Pruned	With 0's	12	0.7211414	0.5808969

The confusion matrix for the pruned tree is shown below. The results appear similar to the initial CART

model, but with slightly lower overall accuracy at 0.7211414.



Pruning the tree produces a much more compact tree, less complex tree - but only about 1/3 to 1/2 of the orders will "make the cut" and be included in the pruned CART model.

The code below allows you to select a random row from the data and try to predict the outcome using our pruned CART model.

This code will produce one random prediction using the rpart pruned CART model:

```
## [1] "Chiroptera"
```

```
## [1] "Chiroptera"
```

Random Forests

Having established the possible benefits of a CART approach, A random forests ensemble method may provide even better results and a more robust model. One challenge to using this method with the data we have, is that random forests are sensitive to NA values. With NA values in the data set, the random forest will apply complete case analysis. Even though there are 29 different orders in the training data, since many of them have at least one NA value on every row, fewer than half of these orders would *ever* be selected by the model. We will take a look at the difference between a data set with NA's, 0's or other imputed values.

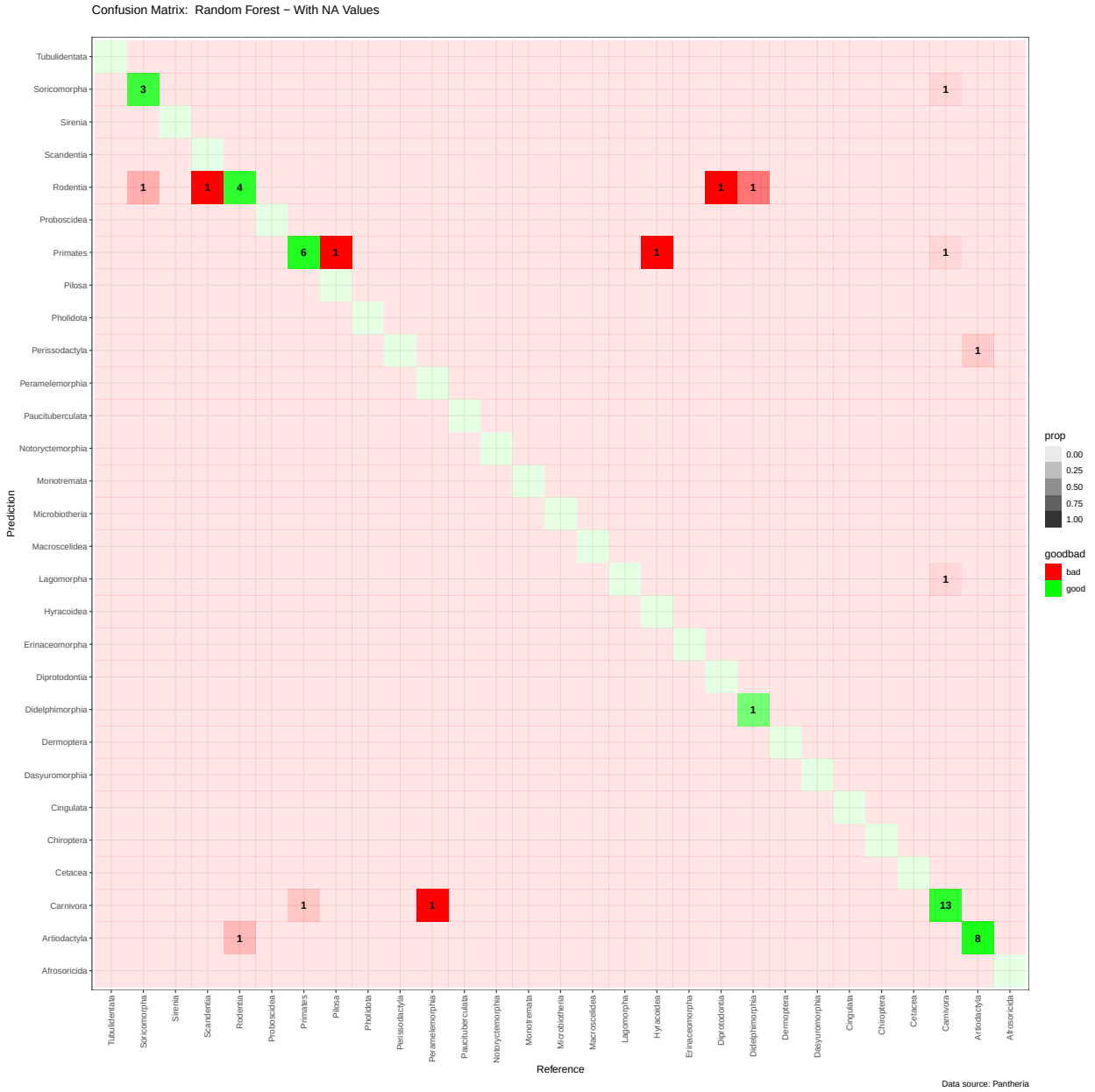
As noted previously - there are some non-random patterns in the missingness of data - some of the columns are applicable only to land based, flying, or marine mammals. During model construction we will focus on the common columns for all mammals. In the final model construction, we will experiment with adding available case analysis to enable use of more specialized columns like forearm length and geo coordinates that are not as universally recorded or where special handling will be required.

We have set the training set and test set to use. This is the data set with NA values.

The model is now trained. We will now apply a prediction on the test data with NA's. This is expected to produce a limited number of actual predictions since it will only be able to train using complete case analysis.

model	dataset	Predicted	Accuracy	F1_Mean
1- Baseline	With 0's	1	0.3852140	0.5561798
2- CART - Unpruned	With 0's	16	0.7496757	0.6891504
3- CART - Pruned	With 0's	12	0.7211414	0.5808969
4- Random Forest	With NA's	6	0.7291667	0.7516083

This model can only produce 48 predictions out of 771 challenges.



The random forest model with NA's results in very high accuracy, but there are many NA's in the predictions. This is not a very practical approach if you want to produce a prediction for every row of test data. To improve the number of outcomes predicted, we will experiment with various strategies to impute data, in order to "fill in blanks", and attempt to balance the classes to encourage the model to predict more minority classes.

Imputing 0's

There are missing values on many rows, relative to the overall data set. (NA's were originally denoted in the data set as -999, these values have been converted to an R standard *NA*) The sparseness of this data has the potential to skew any statistical analysis and presents a challenge to development of a classification tree model. There are several options available to handle the missing data.

Imputing 0 is the simplest possible approach - to fill in any values that are missing from the data with a 0.

Imputing 0's is a crude way of assigning a value that represents missing data. The fact that data is missing may be predictive in the Pantheria data, but this is not necessarily the case for data collected through other outside sources.

Imputing the mean or median, or a best-guess predicted value may also be a way to fill in the gaps in the training data. Every approach to imputation has some advantages and drawbacks. Some methods are easier to implement, but may inaccuracies in the training data. Other methods may be more complex, but produce more accurate imputations. It is ultimately a choice in the final implementation, how to apply imputation to the data, depending on the desired model performance.

In this data set - missingness is somewhat of a predictor - certain fields are set only on certain orders as previously discussed. Chiroptera are the only order with arm lengths measured, Cetacea and Sirenia do not have geocodes and related fields. Missingness can be a valuable hint as to the correct class, but it is also potentially overfitting the model by assuming that real-world challenge data is going to have the same patterns of missingness.

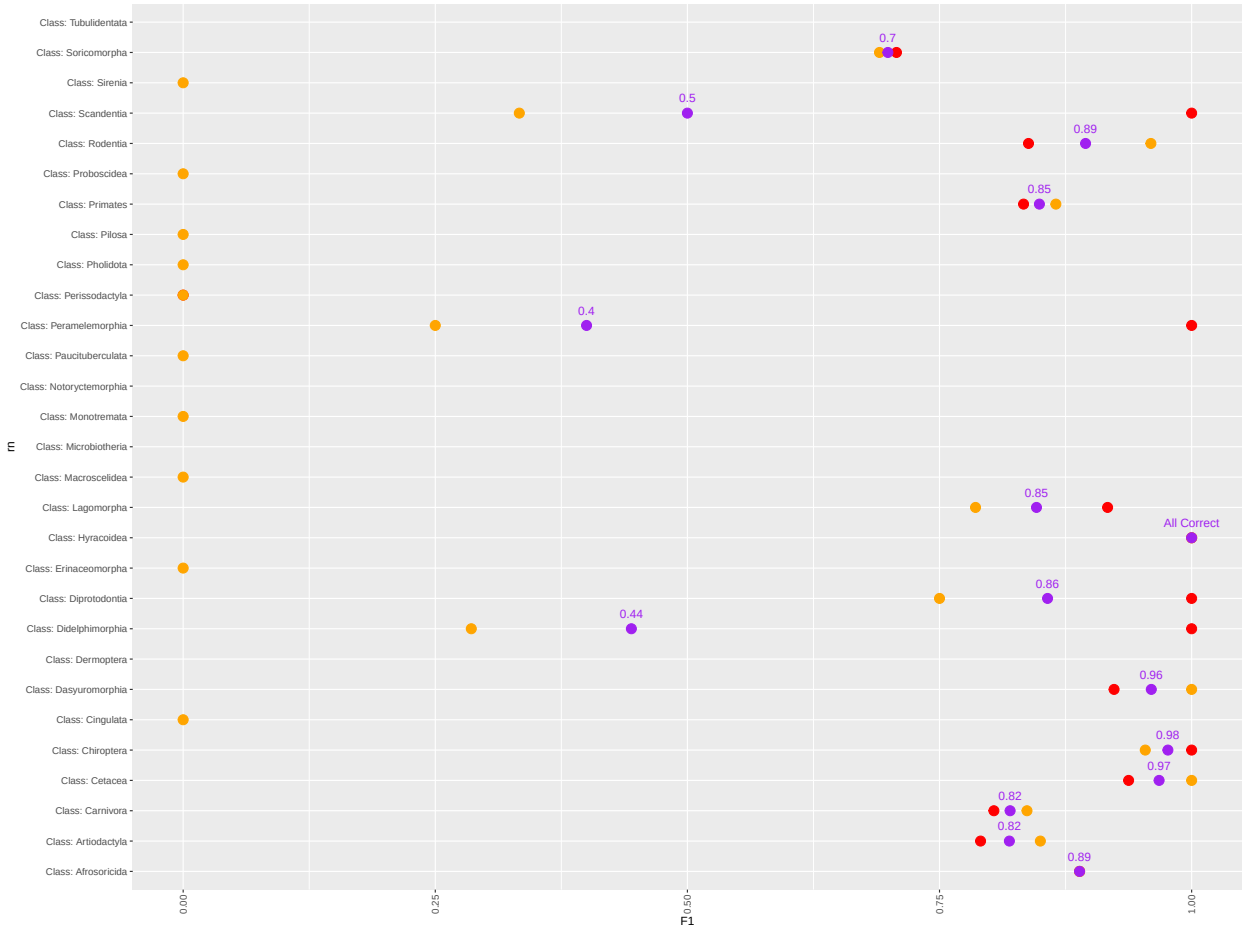
We will start by imputing 0 for all missing values. This is similar to using an available case analysis approach - by adding 0 the RF will be able to use all the rows in the training data, since it will not encounter any columns for which there are NA values.

Using this method will result in many false pieces of information in the training data, like animals with 0 mass. But applying 0 to both the training and test data means that there will be many cases where a 0 in a training data column will match a 0 in a test row, because both were missing the same column.

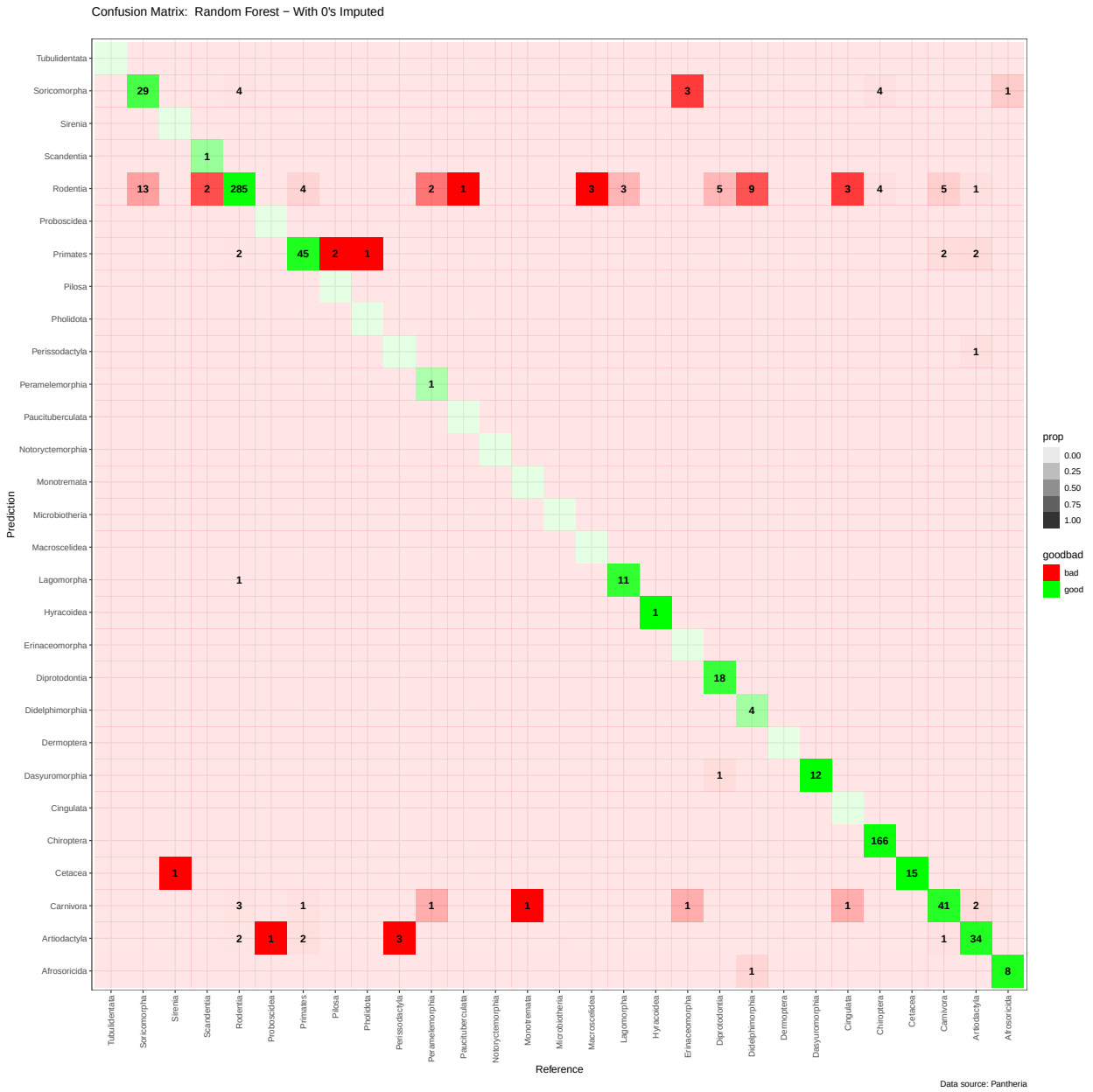
After experimenting with the imputation of 0's, We will try various other imputation strategies throughout the model building section to try to meet our stated goals and make the final model less reliant on matching these patterns of missingness.

model	dataset	Predicted	Accuracy	F1_Mean
1- Baseline	With 0's	1	0.3852140	0.5561798
2- CART - Unpruned	With 0's	16	0.7496757	0.6891504
3- CART - Pruned	With 0's	12	0.7211414	0.5808969
4- Random Forest	With NA's	6	0.7291667	0.7516083
5- Random Forest	With 0's	15	0.8702983	0.7948527

Here is another way to visualize the results. Red dots indicate precision, orange indicates recall, and the F1 score is the purple dot. The F1 Score is stamped on the chart.



The random forest model with 0 imputed into NA's results in an overall accuracy of 0.8702983. This has increased the number of classes predicted and resulted in fairly accurate predictions across many of the classes. However the imputation of 0 in training and test data may be overfitting the model by expecting the same pattern of missingness.



Taking a few examples we can see the most common class, Rodentia has been correctly identified the most times. Some rodents were misidentified a variety of different incorrect orders.

A random forest model that imputes 0 to every NA value is shown. This model is quite accurate, with 0.8702983 overall accuracy. It correctly identifies the majority of rows across 18 of the 25 orders in the test data.

Oversampling

Not only does this data suffer from sparseness, but it also has very imbalanced classes. By oversampling the minority classes, we can provide enough observations to appear more frequently in the training data sets and increase the chance that a minority class is selected by the model.

```r
#oversample function will oversample any rows in traindDS
#classes of grouping with less than min_group, up to max_draws
oversample <- function(trainDS, min_group, max_draws, grouping){

  trainDS2 <- trainDS

  minority_ord <- trainDS %>%
    group_by(!!sym(grouping)) %>%
    summarize(group_count =  n()) %>%
    filter(group_count <= min_group)

    minorityrows <- trainDS %>% inner_join(minority_ord, by=grouping)
    minorityrows <- minorityrows %>% select(-group_count)
    minorityrows

    trainDS2 <- trainDS2 %>% bind_rows(minorityrows %>% sample_n(max_draws,replace=TRUE))

  trainDS2
}
```

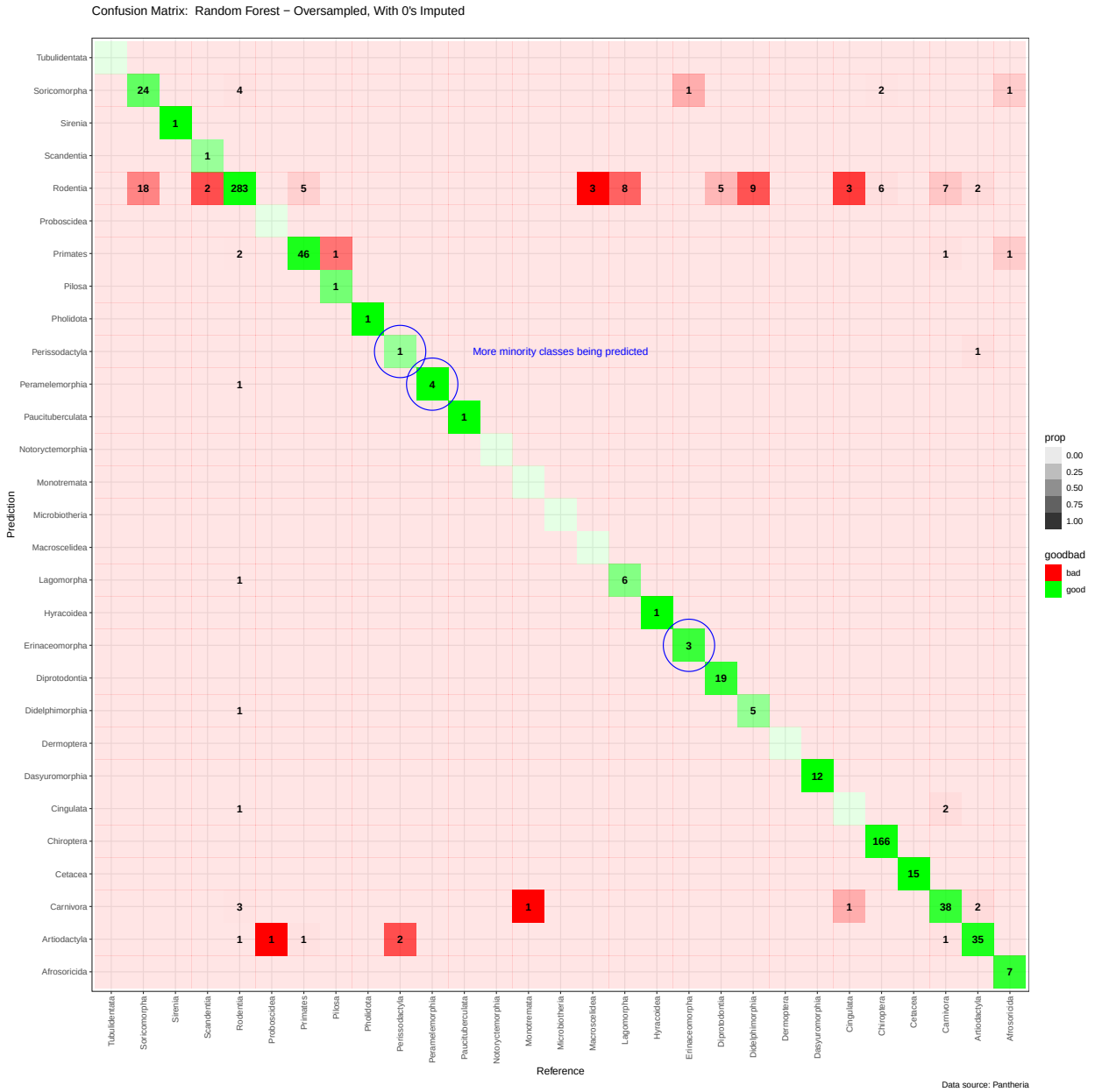
This oversample function will draw random rows from the minority classes rows up to 5000 new rows. The number of rows to add via oversampling is subject to tuning. Oversampling to about 2-3 times the original number of rows was found to provide the best results.

taxo_order	group_count
Afrosoricida	35
Artiodactyla	162
Carnivora	196
Cetacea	60
Chiroptera	695
Cingulata	16
Dasyuromorphia	47
Dermoptera	2
Didelphimorphia	54
Diprotodontia	94

taxo_order	group_count
Afrosoricida	35
Artiodactyla	162
Carnivora	196
Cetacea	60
Chiroptera	695
Cingulata	738
Dasyuromorphia	47
Dermoptera	80
Didelphimorphia	54
Diprotodontia	94

Minority classes like Cingulata and Dermoptera which had only a few rows in the first table have significantly increased the number of rows of data in the second table. We will now train with the oversampled data.

model	dataset	Predicted	Accuracy	F1_Mean
1- Baseline	With 0's	1	0.3852140	0.5561798
2- CART - Unpruned	With 0's	16	0.7496757	0.6891504
3- CART - Pruned	With 0's	12	0.7211414	0.5808969
4- Random Forest	With NA's	6	0.7291667	0.7516083
5- Random Forest	With 0's	15	0.8702983	0.7948527
6- Random Forest	With 0's, Oversampled	21	0.8690013	0.8193970



After oversampling the minority classes, the model attains an accuracy of 0.8690013, and it does better at finding the less common classes. For example - Lagomorpha, Erinaceomorpha, etc. are more consistently chosen. In this model, ~ 21 of the orders are correctly identified at least once. So oversampling seems to have the expected effect of revealing more of the rare classes that would be missed in an imbalanced data set.

Imputing a Grouped Mean

It should be possible to impute a more realistic value than 0 for the missing values, which could make the model more robust and able to handle data entries that don't exactly match the combination of columns recorded in Pantheria. This imputation would be expected to reduce the precision somewhat, as it is adding a lot of new "theoretical" data. The objective would be that the theoretical data could be correct, and could closely match actual test values passed in to the model.

The imputed value will be an estimate based on the mean of each genus, family or order. Imputation increases the homogeneity of the data set, and will consequently underestimate the standard deviation of any statistical analysis. So the mean data should be considered speculative - but that's OK - we are only using it for training the model. It should make the model more robust and adaptable to different sets of input parameters which may or may not match the data provided by Pantheria.

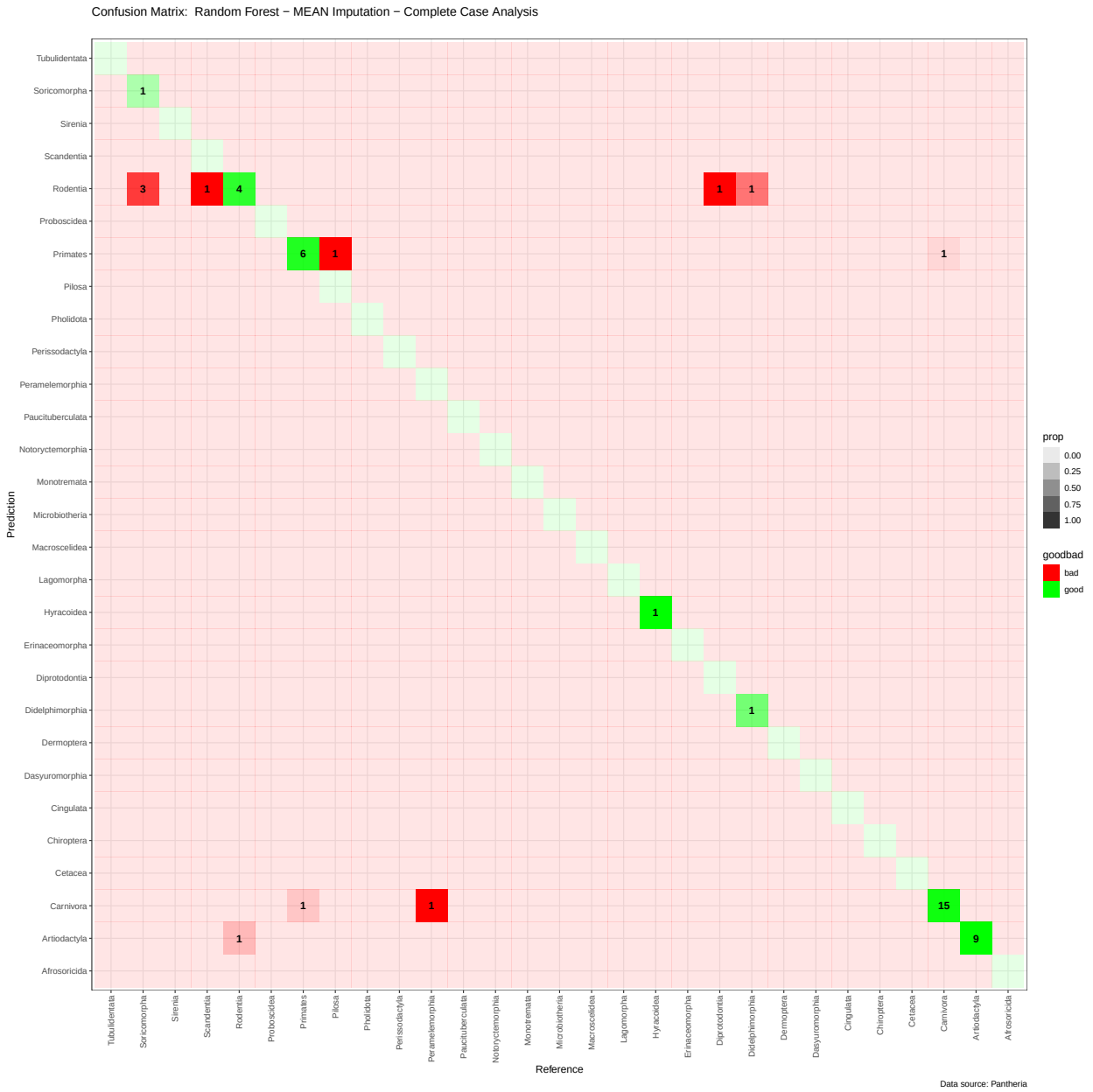
To impute mean values - we will create a copy of the training data set that replaces missing values with estimated means based on the taxonomic classes assigned. This code will first attempt to get an average for each column that's missing, by grouping first at the genus, then the family, then the order, and finally from the entire Mammalia class.

taxo_order	taxo_family	taxo_genus	taxo_species	mass_grams	headlen_mm
Afrosoricida	Chrysochloridae	Chrysospalax	villosus	108.35000	150.0000
Afrosoricida	Chrysochloridae	Neamblysomus	julianae	21.99000	100.8400
Afrosoricida	Tenrecidae	Microgale	dryas	40.00000	88.9300
Afrosoricida	Tenrecidae	Micropotamogale	lamottei	69.59000	160.0000
Afrosoricida	Chrysochloridae	Chrysochloris	visagiei	36.72000	106.0000
Afrosoricida	Tenrecidae	Microgale	talazaci	43.07000	150.0000
Afrosoricida	Tenrecidae	Micropotamogale	ruwenzorii	109.13000	160.0000
Afrosoricida	Tenrecidae	Potamogale	velox	670.99000	320.0000
Afrosoricida	Tenrecidae	Microgale	principula	10.20000	88.9300
Afrosoricida	Chrysochloridae	Calcochloris	tytonis	44.49818	109.4627

The MEAN imputation has added estimates based on a grouped mean in every missing cell for all common values. We can now train a new random forest using this data and see how well it performs.

This version uses a complete case analysis similar to model 4. It produces very few predictions, but they are more accurate than a simple random forest with NA's. This implies that the MEAN data is helping the accuracy when it can match rows of data in the test data with imputed mean values in the training data.

model	dataset	Predicted	Accuracy	F1_Mean
1- Baseline	With 0's	1	0.3852140	0.5561798
2- CART - Unpruned	With 0's	16	0.7496757	0.6891504
3- CART - Pruned	With 0's	12	0.7211414	0.5808969
4- Random Forest	With NA's	6	0.7291667	0.7516083
5- Random Forest	With 0's	15	0.8702983	0.7948527
6- Random Forest	With 0's, Oversampled	21	0.8690013	0.8193970
7- Random Forest	MEAN Imputation with NA's	7	0.7708333	0.7509228



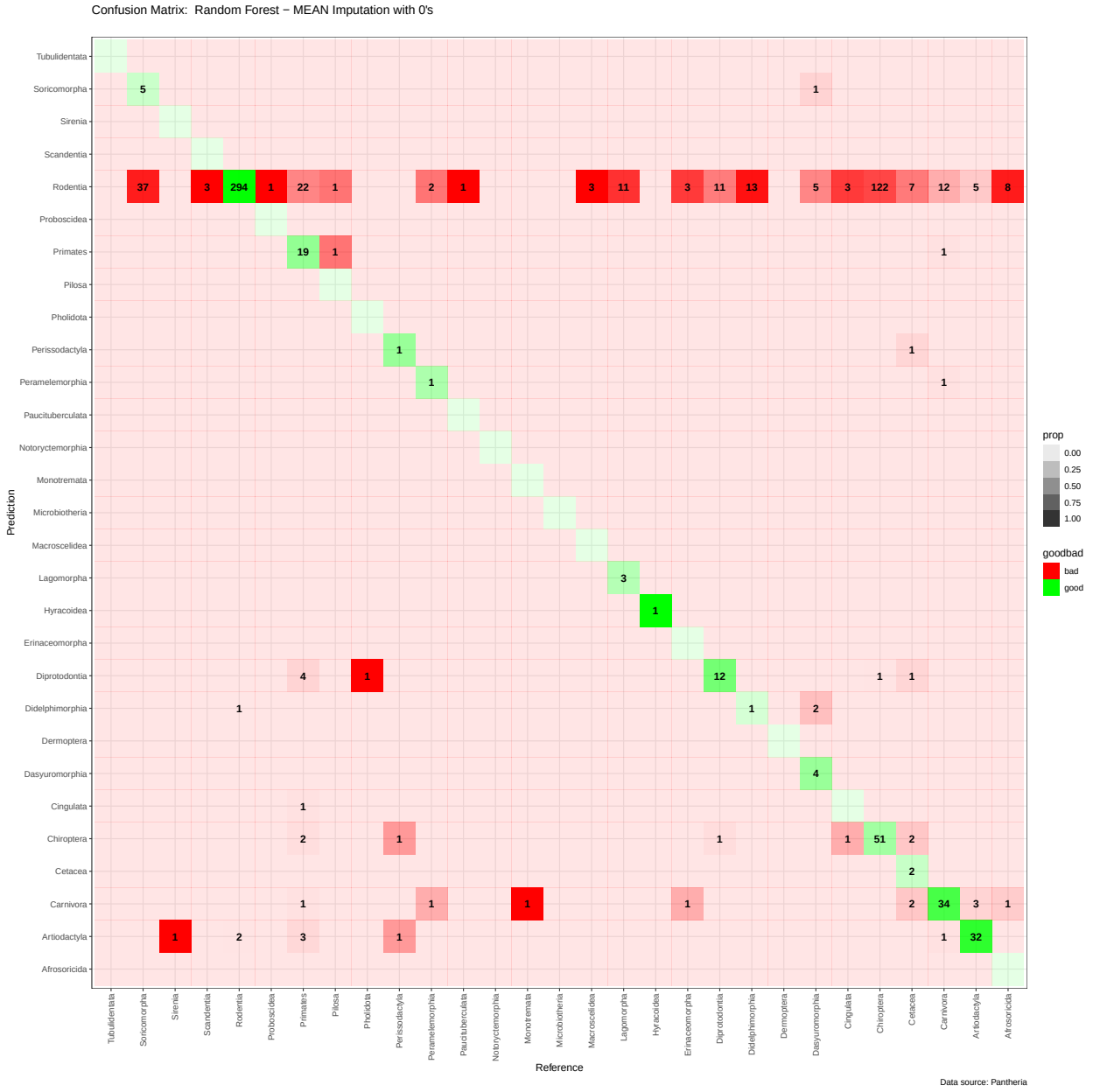
Changing the NA's to 0 in the test data could increase outcomes - but it will probably reduce accuracy, since there will be many cases where a 0 imputed in the test data follows a decision tree rule based on a imputed mean estimate in the training data, and this will "throw off" the decision tree.

We have imputed 0 to the test data to try this out.

model	dataset	Predicted	Accuracy	F1_Mean
1- Baseline	With 0's	1	0.3852140	0.5561798
2- CART - Unpruned	With 0's	16	0.7496757	0.6891504
3- CART - Pruned	With 0's	12	0.7211414	0.5808969
4- Random Forest	With NA's	6	0.7291667	0.7516083
5- Random Forest	With 0's	15	0.8702983	0.7948527
6- Random Forest	With 0's, Oversampled	21	0.8690013	0.8193970
7- Random Forest	MEAN Imputation with NA's	7	0.7708333	0.7509228

model	dataset	Predicted	Accuracy	F1_Mean
8- Random Forest	MEAN Imputation with 0's	14	0.5966278	0.4909618

As expected, the accuracy goes way down, because the imputed 0's in test data and imputed mean in the training data don't match well.



The challenge here, is that the test data can't be imputed in the same way as the training data- we don't know the genus, family or order of the test rows - so we can't impute based on these unknown factors. Any comparison between test data with 0's and *imputed* training data with estimates will lead to significant mismatches in the data.

However - using *imputed* data for training combined with a more complex, *available case* model may produce a "best of both worlds" result - with our informed guess for all fields in the training data being used as the basis of predictions - but ONLY training on the same combination of fields that are supplied in the test data.

The most obvious benefit of this approach, is that since we know that marine mammals never have geography data, and because only bats have forearm length - an available case model will allow use of these predictors, but ONLY when provided as part of the test data. It will also use of test data that has only a portion of the predictors provided. An available case model should allow more of an apples-to-apples comparison if a row in the test data has the same fields as a similar species in the training data- and even if some predictors are not present in the training data- the available case model will adapt by using the estimated mean values for those fields only.

Model Analysis

After all the foregoing model tests, the method that appears to give the greatest overall number of outcomes, and highest accuracy against our 20% internal test data is the oversampled data with 0 imputed.

The accuracy of this model is about the same as that of other models, but it is also producing a broader range of possible outcomes due to the oversampling of minority classes.

Other imputation strategies and use of available case analysis strategies may make this model more robust, at the cost of some accuracy.

We will now switch to the final encapsulation and testing of the model against the hold-out data.

Encapsulation of the Model

Now that the model has been developed, we will encapsulate the models into self-contained functions.

Predict Order - Basic Model

Here is a basic model that will predict the order and impute 0 if indicated and/or apply oversampling.

```
#SIMPLE NON-RECURSIVE MODEL TO PREDICT ORDER
predictorder <- function(trainDS,testDS,impute,oversamplemin,oversamplemax){
  set.seed(1, sample.kind = "Rounding")

  #IMPUTE 0 IF INDICATED
  if(impute == 'zero'){
     trainDS[is.na(trainDS)]<-0
     testDS[is.na(testDS)]<-0
  }

  #oversample if indicated
  if(oversamplemin > 0){
    trainDS <- oversample(trainDS,oversamplemin,oversamplemax,'taxo_order')
  }

  trainDS <- trainDS %>% select(-taxo_family,-taxo_genus,-taxo_species)
  testDS <- testDS %>% select(-taxo_family,-taxo_genus,-taxo_species)

  #set.seed(1, sample.kind = "Rounding")
  # Random Forest prediction of order data
  library(randomForest)
  rf_fit <- randomForest(
    as.factor(taxo_order)~.,
```

```
    data=trainDS,
    na.action=na.roughfix
    )

  ##---------------------------------------
  prx <- predict(object=rf_fit,testDS,type="class")
  tz_actual <- tz_test %>% mutate(taxo_order = as.factor(taxo_order)) %>% pull(taxo_order)

  cm <- confusionMatrix(data=prx,reference=tz_actual)

  returnobj <-list(
                  confusionmatrix=cm,
                  predictions=prx
                  )

  class(returnobj) <- "rfresults"
  returnobj
}
```

To explore a more sophisticated implementation of the model, that may be more robust and practical - we will now build a version that will make predictions to the family and genus ranks, uses mean imputation, available case analysis, and oversampling.

Predict Genus - Available Case Model

This version of the model uses only available cases in the training data having all of the same predictors set. The MEAN data will have values set on every row for all predictors, either factual or estimated. The model will train a random forest and make a prediction for one test row at a time, using ONLY the list of columns specified in that one row of test data. It will then do the same process for the family, and the genus.

This is a bit complex to accomplish. Here's how it will work:

Round 1 - Order (least specific)

- First, we sample 1 row of data which includes, family and order and genus.
- Create a Training Set that removes family & genus columns leaving only the order
- Determine the columns in the test data & include only those fields in training data
- Fit the RF Model for the order rank
- Make a prediction for order

Round 2 - Family (more specific)

- Filter the original training data to include only the family
- Create a Training Set that includes only the predicted order, with only the family column and predictors
- Determine the columns in the test data & include only those fields in training data
- Fit the RF Model for the family rank
- Make a prediction for family

Round 3 - Genus (most specific)

- Filter the original training data to include only the genus
- Create a Training Set that includes only the predicted family, with only the genus column and predictors

- Determine the columns in the test data & include only those fields in training data
- Fit the RF Model for the genus rank
- Make a prediction for genus

This approach enables us to predict across more than 53 outcome classes, which is the upper limit of the random Forest package. Since the correct *order* will be predicted about 90% of the time, this would be the maximum expected accuracy when predicting at the *family* or *genus* level. If the first guess was wrong, the second must be wrong. However if it predicts the order right in the first round, then it should have a pretty good chance of getting the family right on the second and third step, etc. as it will use a more focused inter-class data set to make this second and third layer of prediction.

```
#function to extract columns
extract_columns <- function(data, desired_columns) {
    extracted_data <- data %>%
        select_(.dots = desired_columns)
    return(extracted_data)
}
#ADVANCED RECURSIVE MODEL TO PREDICT ORDER, FAMILY & GENUS
predictgenusac <- function(trainDS,testRow,last_a){

  predicted_order <- ''
  predicted_family <- ''
  predicted_genus <- ''

  ##------------------------------------
  #SETUP - LEAVE ONLY ORDER DATA
  train_order <- trainDS %>% select(-taxo_family, -taxo_genus, -taxo_species)
  test_order <- testRow %>% select(-taxo_family, -taxo_genus, -taxo_species)

  #remove the reference class for the purpose of making a prediction
  test_order_pred <- test_order %>% select(-taxo_order)
  onerow <- testRow

  #Select a list of names of the columns that are in the test set
  usecolumns <- colnames(test_order_pred)[colSums(!is.na(test_order_pred)) > 0]
  test_columns <- length(usecolumns)

  #print(paste("Test Columns:",test_columns))
  #print(paste("Row:",last_a))

  #Extract those columns only from the training set
  train_extracted <- extract_columns(train_order,usecolumns)
  test_extracted <- extract_columns(onerow,usecolumns)

  #Reconstruct the training table with class and predictors only
  train_order <- train_order %>% select(taxo_order) %>% bind_cols(train_extracted)
  train_order <- train_order %>% drop_na()
  train_order$taxo_order <- droplevels(train_order$taxo_order)



  #---------------------------------------
  #PREDICT ORDER
  library(randomForest)
```

```
rf_fit_order <- randomForest(
  as.factor(taxo_order)~.,
  data=train_order,
  na.action=na.roughfix
  )

prx <- predict(object=rf_fit_order,
               test_order_pred,
               type="class")

predicted_order <- prx
predicted_order <- as.character(predicted_order)
#----------------------------------------
#CHECK FOR SUB FAMILES

# print(paste("Predicted Order:",predicted_order))

 sub_family <- trainDS %>%
  filter(as.character(taxo_order) == as.character(predicted_order)) %>%
 group_by(taxo_family) %>% summarize(groupcount = n())

 num_fams <- count(sub_family)$n

 #print(paste("num_fams:",num_fams))

 if(num_fams == 1){
   predicted_family <- sub_family %>% pull(taxo_family)
   predicted_family <- as.character(predicted_family)



    }else{


 ##-------------------------------------
  #SETUP FAMILY DATA
    train_family <- trainDS %>%
    filter(as.character(taxo_order) == as.character(predicted_order)) %>%
    select(-taxo_order, -taxo_genus, -taxo_species) %>%
    mutate(taxo_family = as.character(taxo_family))  %>%
    mutate(taxo_family = as.factor(taxo_family))

    #Leave only Family Class
    test_family <- testRow %>% select(-taxo_order, -taxo_genus, -taxo_species)

    #Leave only predictors
    test_family_pred <- test_family %>% select(-taxo_family)

    #AVAILABLE CASE ANALYSIS
    #Select the columsn that in the test set
    usecolumns <- colnames(test_family_pred)[colSums(!is.na(test_family_pred)) > 0]

    #Extract those columns only from the training set
```

```
    train_extracted <- extract_columns(train_family,usecolumns)
    test_extracted <- extract_columns(onerow,usecolumns)

    #Reconstruct the training table with class and predictors only
    train_family <- train_family %>% select(taxo_family) %>% bind_cols(train_extracted)
    train_family <- train_family %>% drop_na()
    train_family$taxo_family <- droplevels(train_family$taxo_family)

    if (nrow(train_family)>1 ){
        #----------------------------------------
        #PREDICT FAMILY
        rf_fit_family <- randomForest(
        as.factor(taxo_family)~.,
        data=train_family,
        na.action=na.roughfix
        )

        prx <- predict(object=rf_fit_family,test_family_pred,type="class")

        predicted_family <- prx
        predicted_family <- as.character(predicted_family)

    }else{

       predicted_family <- train_family %>% pull(taxo_family)
      predicted_family <- as.character(predicted_family)
      }
    }

#print(paste("Predicted Family:",predicted_family))

    #----------------------------------------
    #CHECK FOR SUB GENUS

    sub_genus <- trainDS %>%
    filter(as.character(taxo_order) == as.character(predicted_order) & as.character(taxo_family) == as
    group_by(taxo_genus) %>% summarize(groupcount = n())
    num_genus <- count(sub_genus)$n

    #print(paste("num_genus:",num_genus))

    if(num_genus == 1){
       predicted_genus <- sub_genus %>% pull(taxo_genus)
       predicted_genus <- as.character(predicted_genus)

    }else{

        ##---------------------------------------
        #SETUP GENUS DATA

       #leave genus & Predictors only in training data
        train_genus <- trainDS %>%
        filter(as.character(taxo_order) == as.character(predicted_order) & as.character(taxo_family) =
```

```
        select(-taxo_order, -taxo_family, -taxo_species)  %>%
        mutate(taxo_genus = as.character(taxo_genus))  %>%
        mutate(taxo_genus = as.factor(taxo_genus))

        #leave predictors and genus class only
        test_genus <- testRow %>%
        select(-taxo_order, -taxo_family, -taxo_species)

        #leave predictorsonly
        test_genus_pred <- test_genus %>% select(-taxo_genus)

        #AVAILABLE CASE ANALYSIS
        #Select the columsn that in the test set
        usecolumns <- colnames(test_genus_pred)[colSums(!is.na(test_genus_pred)) > 0]

        #Extract those columns only from the training set
        train_extracted <- extract_columns(train_genus,usecolumns)
        test_extracted <- extract_columns(onerow,usecolumns)

        #Reconstruct the training table with class and predictors only
        train_genus <- train_genus %>% select(taxo_genus) %>% bind_cols(train_extracted)
        train_genus <- train_genus %>% drop_na()
        train_genus$taxo_genus <- droplevels(train_genus$taxo_genus)


        if (nrow(train_genus)>1 ){
            #----------------------------------------
          #PREDICT GENUS
          rf_fit_genus <- randomForest(
          as.factor(taxo_genus)~.,
          data=train_genus,
          na.action=na.roughfix
          )

          prx <- predict(object=rf_fit_genus,test_genus_pred,type="class")

          predicted_genus <- prx
          predicted_genus <- as.character(predicted_genus)
          predicted_genus
       }else{
          predicted_genus <- train_genus %>% pull(taxo_genus)
          predicted_genus <- as.character(predicted_genus)
       }



 }

#print(paste("Predicted Genus:",predicted_genus))


returnobj <- list(
  order=predicted_order,
```

```
    family=predicted_family,
    genus=predicted_genus,
    test_cols=test_columns
  )

  returnobj


}
```

This is the container for the actual model that runs through the test data one row at a time.

```
#ADVANCED MODEL CONTAINER FUNCTION
acgenusmodel <- function(trainDS,testDS,impute,oversamplemin,oversamplemax){
#create a table of the results of bernoulli trails using the hold-out data to train the recursive model
#the recursive model must run one row at a time since each row it chooses spawns a new model training p
trials <- data.frame()
attempts <- seq(1:nrow(testDS))
set.seed(1, sample.kind = "Rounding")

#impute zero if indicated
if(impute == 'zero'){
     trainDS[is.na(trainDS)]<-0
     testDS[is.na(testDS)]<-0
}
#impute means if indicated
if(impute == 'mean'){
    trainDS <- imputeMeans(trainDS)
}

#set the training to the entire dataset
trainDS_recursive <- trainDS

if(oversamplemin > 0){
    trainDS_recursive_os <- oversample(trainDS_recursive,oversamplemin,oversamplemax,'taxo_order')
}else{
    trainDS_recursive_os <- trainDS_recursive
}

#make some predictions
for(a in attempts){

  last_a <- a
  onerow <- testDS %>% slice(a:a)
  predicted <- predictgenusac(trainDS_recursive_os,onerow,last_a)

  order_match <- (as.character(predicted$order) == as.character(onerow$taxo_order))
  family_match <- (as.character(predicted$family) == as.character(onerow$taxo_family))
  genus_match <- (as.character(predicted$genus) == as.character(onerow$taxo_genus))

  newrow <- data.frame(pred_order=as.character(predicted$order),
                       pred_fam=as.character(predicted$family),
                       pred_genus=as.character(predicted$genus),
                       correct_order=as.character(onerow$taxo_order),
```

```
                        correct_fam=as.character(onerow$taxo_family),
                        correct_genus=as.character(onerow$taxo_genus),
                        correct_species=as.character(onerow$taxo_species),
                        test_cols=as.character(predicted$test_cols),
                        order_correct=order_match,
                        family_correct=family_match,
                        genus_correct=genus_match
                        )

  trials <- trials %>% bind_rows(newrow)

}

order_acc2 <- mean(trials$order_correct)
family_acc2 <- mean(trials$family_correct)
genus_acc2 <- mean(trials$genus_correct)

trials <- trials %>% left_join(order_info, by = c("pred_order" = "taxo_order"))
trials <- trials %>% left_join(order_info, by = c("correct_order" = "taxo_order"))

trials
}
```

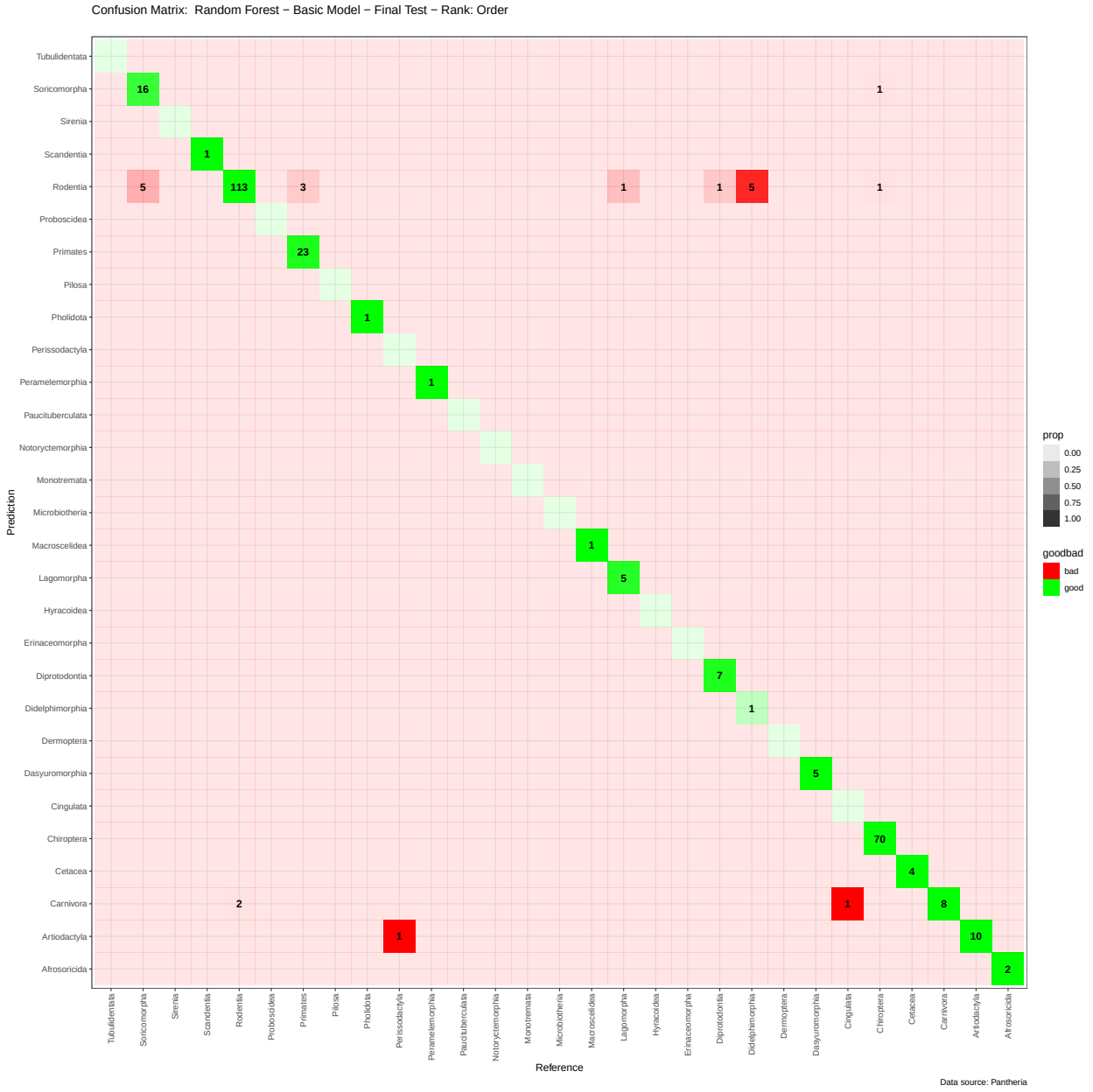
Results

Calling the Basic Model

The **predictorder** function will run the basic order-level prediction model with 0 imputed and all data columns used.

Assessing the Basic Model

We have run the basic model against our hold-out data set. Now we will examine how the basic model performed.



model	dataset	Predicted	Accuracy	F1_Mean
1- Baseline	With 0's	1	0.3852140	0.5561798
2- CART - Unpruned	With 0's	16	0.7496757	0.6891504
3- CART - Pruned	With 0's	12	0.7211414	0.5808969
4- Random Forest	With NA's	6	0.7291667	0.7516083
5- Random Forest	With 0's	15	0.8702983	0.7948527
6- Random Forest	With 0's, Oversampled	21	0.8690013	0.8193970
7- Random Forest	MEAN Imputation with NA's	7	0.7708333	0.7509228
8- Random Forest	MEAN Imputation with 0's	14	0.5966278	0.4909618
9- Random Forest	Basic Model - Order - FINAL	16	0.9273356	0.9134782

The basic model is able to predict the order with overall accuracy of 0.9273356. This looks like very high accuracy, but as noted earlier - this may be the result of overfitting, particularly with respect to missingness - as it would rely on the patterns of missingness being the same in a real world challenge scenario. We will now examine how the Advanced model compares on the same hold-out data set.

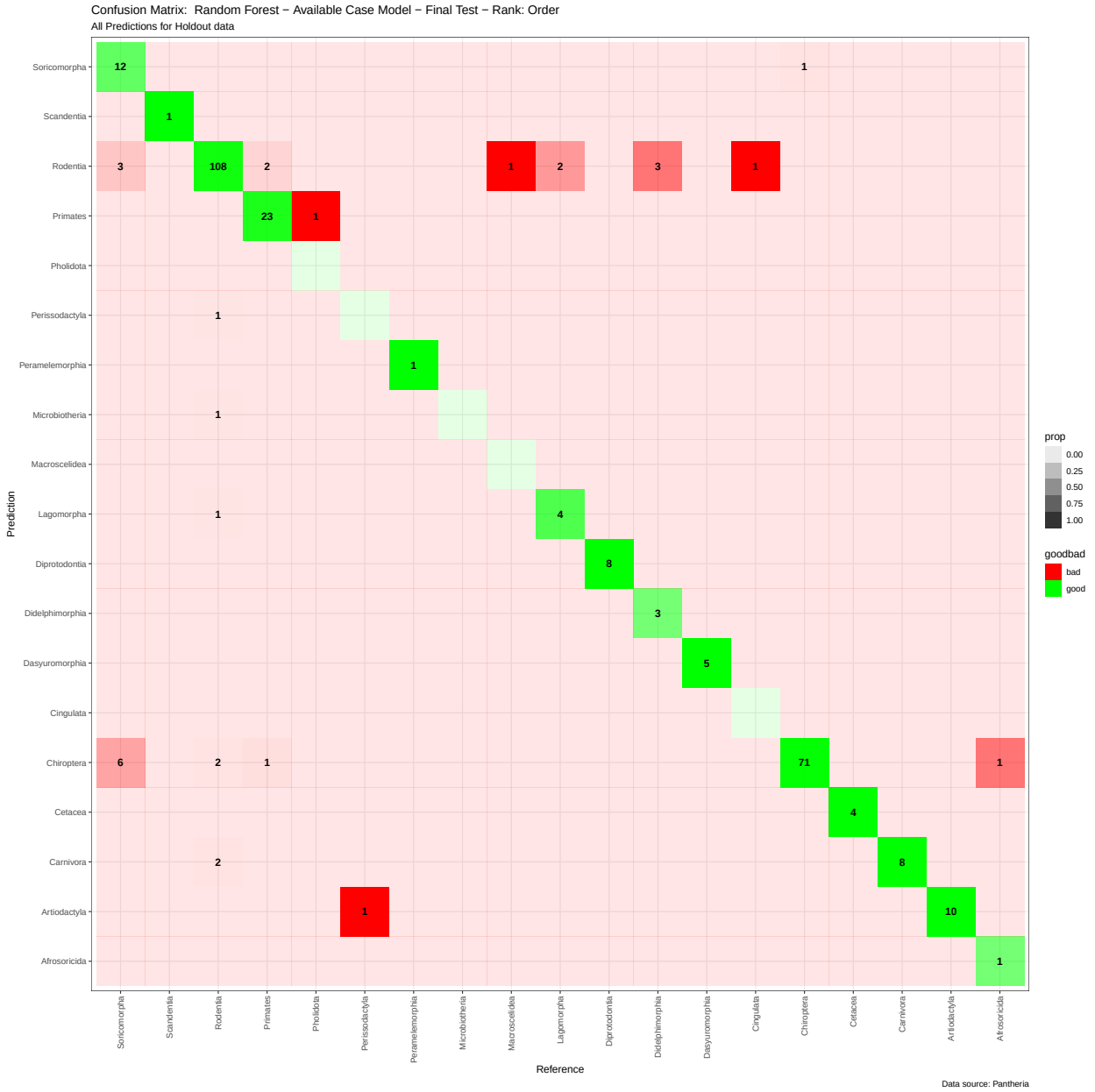
Calling the Advanced Available Case Model

The **acgenusmodel** function will run the final available case model. This model usually takes 10-15 minutes to run against the ~300 rows of hold-out testing data.

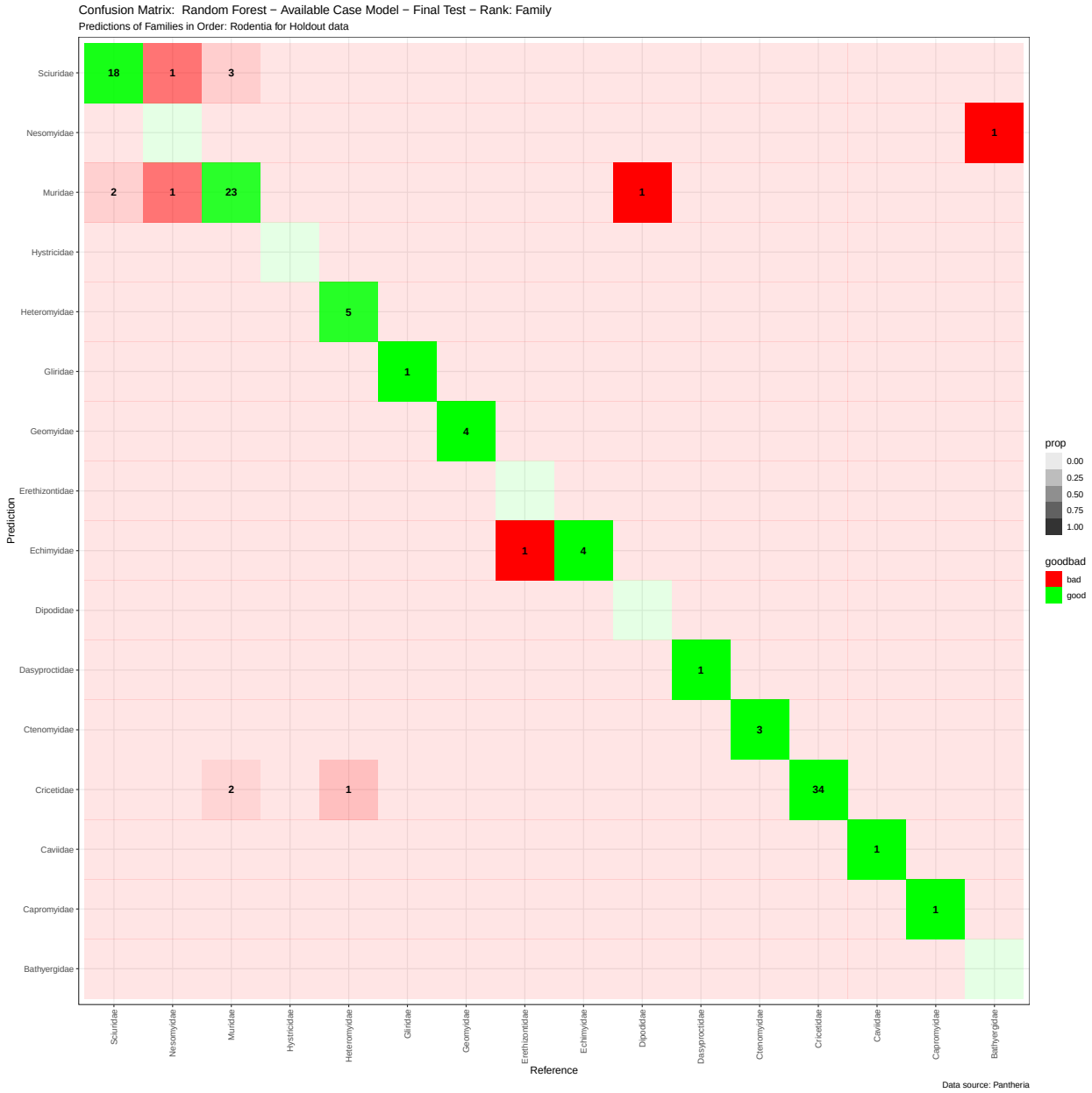
Randomly selecting 10 rows reveals how the model performed on just a few of the predictions. It has done pretty well - making a reasonable guess at both the Order and family level in most cases, with more confusion at the genus level, as expected.

pred_order	pred_fam	pred_genus	correct_order	correct_fam	correct_genus
Chiroptera	Pteropodidae	Pteralopex	Rodentia	Muridae	Uromys
Cetacea	Delphinidae	Lagenorhynchus	Cetacea	Ziphiidae	Mesoplodon
Primates	Cercopithecidae	Cercopithecus	Primates	Cercopithecidae	Cercopithecus
Chiroptera	Pteropodidae	Scotonycteris	Chiroptera	Rhinolophidae	Rhinolophus
Rodentia	Cricetidae	Oecomys	Rodentia	Cricetidae	Tylomys
Soricomorpha	Talpidae	Euroscaptor	Soricomorpha	Soricidae	Chimarrogale
Chiroptera	Vespertilionidae	Phoniscus	Chiroptera	Vespertilionidae	Kerivoula
Rodentia	Sciuridae	Petaurista	Lagomorpha	Leporidae	Lepus
Rodentia	Cricetidae	Peromyscus	Didelphimorphia	Didelphidae	Thylamys
Rodentia	Echimyidae	Proechimys	Rodentia	Echimyidae	Proechimys

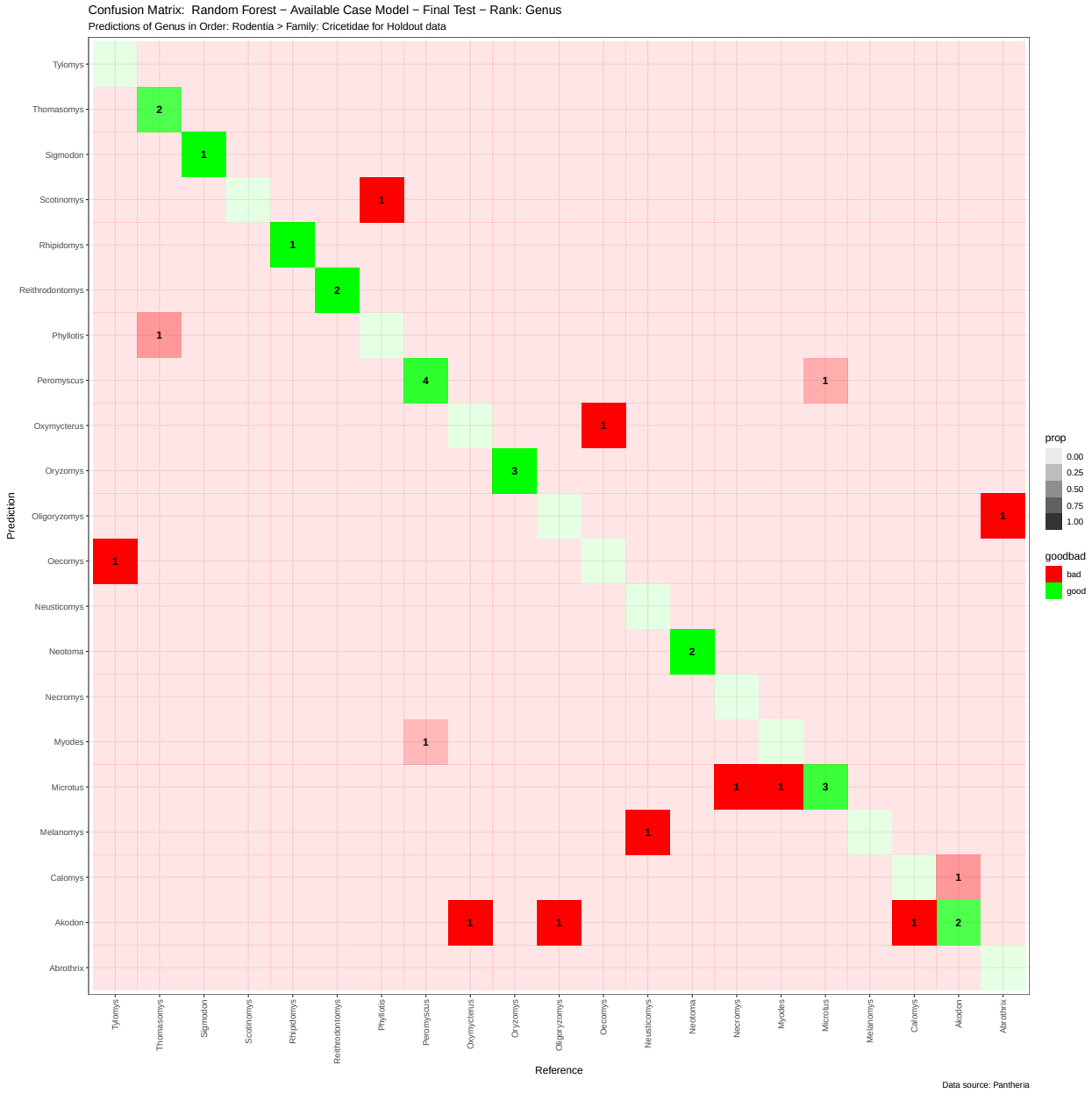
Having examined a few of the raw predictions, we will now use our familiar confusion matrix visualization charts to examine the outcome of the final advanced available case model. Since this model is predicting three different sets of factors, three confusion matrices will be necessary to examine the predictions for the ranks of order, family and genus.



The plot above explores the prediction across all orders. We can drill down to the family rank and observe how the model has performed within one order. We will use Rodentia since it is the order with the most predictions.



And finally we can drill down to the genus rank and observe how the model has performed within one family. We will use Rodentia > Cricetidae since it is the family with the most predictions.



Assessing the Advanced Model

Now we will see how the advanced model has performed across all predictions at each rank.

model	dataset	Predicted	Accuracy	F1_Mean
1- Baseline	With 0's	1	0.3852140	0.5561798
2- CART - Unpruned	With 0's	16	0.7496757	0.6891504
3- CART - Pruned	With 0's	12	0.7211414	0.5808969
4- Random Forest	With NA's	6	0.7291667	0.7516083
5- Random Forest	With 0's	15	0.8702983	0.7948527
6- Random Forest	With 0's, Oversampled	21	0.8690013	0.8193970
7- Random Forest	MEAN Imputation with NA's	7	0.7708333	0.7509228

model	dataset	Predicted	Accuracy	F1_Mean
8- Random Forest	MEAN Imputation with 0's	14	0.5966278	0.4909618
9- Random Forest	Basic Model - Order - FINAL	16	0.9273356	0.9134782
10- Random Forest	Advanced Model - Order - FINAL	16	0.8961938	0.8839294
11- Random Forest	Advanced Model - Family - FINAL	54	0.7439446	0.8404091
12- Random Forest	Advanced Model - Genus - FINAL	169	0.4705882	0.8514535

The final accuracy on the hold-out data using available case analysis, mean imputation, and oversampling applied to the training data, is around 0.8961938 at the order rank; 0.7439446 at the family rank; and 0.4705882 at the genus rank when making predictions against our final hold-out validation data.

The model is predicting 16 different orders out of the 18 distinct orders contained in the hold-out test data; predicting 54 different families out of the 66 distinct families; and predicting 169 different genus outcomes out of 212 in the test data.

Conclusion

It was challenging to construct a model that provided good overall accuracy and made a diverse range of predictions. The final model selected produces fairly high accuracy at the order level, with expected decreases in accuracy as the model predicts at a more specific rank of taxonomy.

One of the challenges of this model was that we did not have very abundant data to begin with, and there were a lot of missing data points. Since each row in Pantheria represents one species, there were a limited number of rows to use, and it was necessary to use a strategic approach to partitioning the data to ensure a good cross section of rows would be randomly selected for training, test, and hold-out data sets.

The Basic model provides a fast and simple way to predict to the order rank. The accuracy is very high, but it is not very useful, and would rely on having test data with similar patterns of missingness. So that model is not very robust and may be somewhat overfitted, especially if it were used with real-world challenge data.

The Advanced available case model solves some of these problems, but at a steep cost in performance and run time. Because it must construct 3 different RF models per prediction, and because the RF model varies depending on the number of data points provided, it can only make one prediction at a time.

Performance Tuning & Next Steps

There is room for further improvement through implementation of more complex imputation strategies on the training data to estimate missing values more precisely. Imputing a grouped mean assumes that taxonomic groups are homogeneous - but imputation based on linear regression may sometimes be more accurate. However the data already contains some extrapolated values - so one must be careful not to over-extrapolate from inferred data. Use of a more sophisticated imputation strategy was beyond the scope of this report - but may be a good subject for further exploration. The MICE package contains many additional methods to explore in this area.

Addition of more data from other sources could also help fortify the training data available from Pantheria. Model tuning through optimization of the Random Forest parameters, or adjustment to the amount or methods of oversampling may also enhance model performance. There are additional packages built around oversampling including the SMOTE package which can provide more ready-made oversampling methods which may also improve performance.

Interactive Demonstration

To demonstrate a potential application of the model, a “Mammal Predictor” Shiny App was developed that utilizes the final, available case model to make a prediction for a user-entered challenge. This interactive shiny app takes various input values and returns the predicted order, family and genus based on any combination of provided inputs.

https://ryancooper.shinyapps.io/ShinyZoo/

Commentary

I hope this report has demonstrated a viable approach for applying random forests to hierarchical taxonomy data, while addressing some of the issues of sparseness and imbalanced classes. It was my goal to demonstrate through this capstone project how I have come to understand and appreciate the R language, and how I have learned to implement lessons from the various courses throughout the HarvardX Data Science Professional Certificate program, including visualization, data wrangling, probability, inference, and machine learning strategies.

This was a fascinating and fun experiment to try to solve several challenges in this data set. The remarkable diversity of life on Earth cataloged through Pantheria and GBIF provides an extensive source of information

- though not always a complete picture. It is quite awe-inspiring to consider that the Pantheria data was derived from over 3000 independent research sources - this project owes a huge debt of gratitude to the many dedicated researchers and data entry technicians who painstakingly compiled this collection of data.

This data set shows off the diversity of nature, and provides ample opportunities to explore the incredible variety of traits observed for the thousands of unique, individual species in the class of Mammalia. The biodiversity of Earth is a precious resource, and it is our responsibility to study, protect and preserve the many unique species with which we share the planet.

Acknowledgements

I would like to thank Dr. Irizarry for the excellent content and lessons provided in this program, and the dedicated TA's and technicians supporting the HarvardX program and EDX elearning platform; A very special Thank You to the student reviewers who took the time to review this project; and lastly, a huge Thank You to my fiancé, Krista, who provided me an immeasurable quantity of patience, understanding and support in this endeavor.

References

Breiman, Leo. (2001). *Random Forests.* Retrieved from Statistics Dept. University of California, Berkeley: https://www.stat.berkeley.edu/~breiman/randomforest2001.pdf

Chen, Chao, (2004). *Using Random Forest to Learn Imbalanced Data.* Retrieved from Statistics Dept. University of California, Berkeley: https://statistics.berkeley.edu/sites/default/files/tech-reports/666.pdf

GBIF Contributors. (2020). *Global Biodiversity Information Facility: Free and open access to biodiversity data.* Retrieved from GBIF: https://www.gbif.org/

Gelman, Andrew; Hill, Jennifer. (2006). *Missing-data imputation, Data Analysis Using Regression and Multilevel/Hierarchical Models.* Cambridge University Press

Irizzary, Rafael. (2020). *Introduction to Data Science.* Retrieved from github page: https://rafalab.github.io/dsbook/

Jones, E., et. al., *PanTHERIA: a species-level database of life history, ecology, and geography of extant and recently extinct mammals.*, (2016). Retrieved from Ecological Society of America: http://esapubs.org/archive/ecol/E090/184/metadata.htm

Rubin, Donald B., (1976, 12). *Inference and missing data.* Biometrika, Volume 63, Issue 3, December 1976, Pages 581–592

Van Buuren, Stef. (2018). *Flexible Imputation of Missing Data*, Retrieved from CRC Press: https://stefvanbuuren.name/fimd/sec-MCAR.html

Wilson, Don E. & Reeder, DeeAnn M.(editors). (2005). *Mammal Species of the World. A Taxonomic and Geographic Reference (3rd ed)*, Retrieved from Johns Hopkins University Press: https://www.departments.bucknell.edu/biology/resources/msw3/

Publisher Information:

Key R Packages:

DPLYR package https://cran.r-project.org/web/packages/dplyr/index.html

GGPlot2 package https://cran.r-project.org/web/packages/ggplot2/index.html

MICE package https://cran.r-project.org/web/packages/mice/index.html

RGBIF Package https://cran.r-project.org/web/packages/rgbif/index.html

RPart Package https://cran.r-project.org/web/packages/rpart/index.html

Using Plot CP in RPart https://www.rdocumentation.org/packages/rpart/versions/4.1-15/topics/plotcp

R Documentation https://www.rdocumentation.org