Predicting Monarchs in Mexico with Machine Learning Models

Keaton Wilson

7/2/2019

Introduction

The basic goal of this project is to generate a Machine Learning model (or models) that predicts hectares in Mexico of Eastern Monarchs (a problematic, but common measure of population success) using citizen-science data and publicly available environmental data. A few challenges:
1. There are only ~ 24-25 years of data.
2. What features/variables do we use?

ML with Small Data

The strategy here is to build a full feature set for our small "real" data, and then use the `synthpop` package to generate a larger data set (~25k observations) of synthetic data based on the properties of the original data set. Then, we'll train ML models on the synthetic data and use the 'real' data to evaluate.

Appropriate Feature Building

The other tricky thing here is building the appropriate variables. Most of this comes out of a blog post by Chip Taylor.

The features I used here are:
1. The date of first sighting in the US
2. Number of sightings above 37º latitude
3. Total number of sightings between March and October
4. Environmental conditions (19 bioclim variables) of the entire eastern corridor
5. Environmental conditions (19 bioclim variables) of the northern part of the territory (above 37º, that will end up creating 'migratory' morphs)

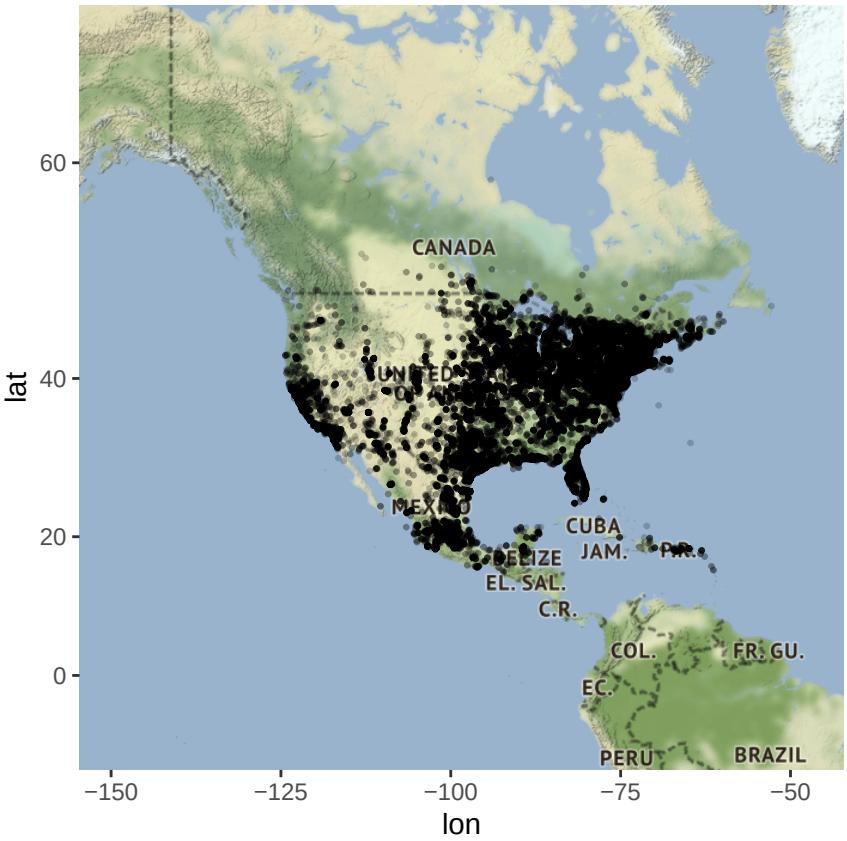
Data Collection and Feature Generation

Data Collection

I used the custom `get_clean_obs` function (a wrapper for spocc) that pulls all (or most, depending on the number of records) of records from GBIF and inat. After some cleaning, filtering and reverse geocoding, we get the table below.

```
## Observations: 85,681
## Variables: 6
## $ prov      <chr> "gbif", "gbif", "gbif", "gbif", "gbif", "gbif", "gbi...
## $ longitude <dbl> -90.12371, -82.54726, -100.26716, -122.04034, -95.57...
## $ latitude  <dbl> 29.93436, 28.03309, 19.58773, 36.95674, 29.97354, 34...
## $ name      <chr> "Danaus plexippus (Linnaeus, 1758)", "Danaus plexipp...
## $ date      <date> 2019-01-01, 2019-01-12, 2019-01-09, 2019-01-25, 201...
## $ locs      <chr> "Gumbel Fountain, 6500 St Charles Ave, New Orleans, ...
```

This is a substantial data set, and we can plot it to look to see if the records make sense.



Ok, these look reasonable. I've taken additional steps not outlined here to do a bit of cleaning on the data before we start building features below.

Feature 1. Date of first sighting

```
library(lubridate)
#Building a dataframe of first sightings
first_us_sighting = monarch_country_sub %>%
  mutate(year = year(date)) %>%
  filter(country == "USA") %>%
  filter(str_detect(locs, "TX|Texas|NM|New Mexico|OK|Oklahoma|AZ|Arizona")) %>% #Restricting to states
  group_by(year) %>%
  summarize(first_sighting = min(date),
            n = n()) %>%
  print(n = 50)
```

```
## # A tibble: 26 x 3
##     year first_sighting     n
##    <dbl> <date>         <int>
##  1  1993 1993-09-24         3
##  2  1994 1994-09-17        16
##  3  1995 1995-09-06        38
##  4  1996 1996-08-08       177
##  5  1997 1997-03-27       771
##  6  1998 1998-02-19         6
```

```
##  7  1999 1999-01-07         2
##  8  2000 2000-09-05         3
##  9  2002 2002-09-29         2
## 10  2003 2003-09-24         2
## 11  2004 2004-04-25         9
## 12  2005 2005-04-12         4
## 13  2006 2006-06-28         9
## 14  2007 2007-04-05        12
## 15  2008 2008-01-01        14
## 16  2009 2009-02-11        14
## 17  2010 2010-06-08         8
## 18  2011 2011-03-01        22
## 19  2012 2012-01-15        26
## 20  2013 2013-01-18        56
## 21  2014 2014-02-23       168
## 22  2015 2015-01-07       415
## 23  2016 2016-01-07      1035
## 24  2017 2017-01-01      1151
## 25  2018 2018-01-09      1968
## 26  2019 2019-01-01       790
```

So, they're really all over the place, but they appear to be getting sooner as time goes on? Interesting. Regardless, we're going to bind them to the master data frame

```
#binding
monarch_ml_df = monarch_ml_df %>%
  left_join(first_us_sighting, by = "year") %>%
  mutate(day_first_sighting = lubridate::yday((first_sighting))) %>%
  dplyr::select(-first_sighting)
```

Feature 2. Number of sightings north of 37º

Adding a feature that is the number of sightings north of 37º divided by/normalized by the total number of sightings for a year

```
monarch_ml_df = monarch %>%
  mutate(year = year(date)) %>%
  group_by(year) %>%
  summarize(n_obs_total = n()) %>%
  right_join(monarch_ml_df, by = "year") %>%
  left_join(monarch %>%
              mutate(year = year(date)) %>%
              filter(latitude > 37) %>%
              group_by(year) %>%
              summarize(n_obs_37 = n()), by = "year") %>%
  mutate(obs_37_norm = n_obs_37/n_obs_total) %>%
  dplyr::select(year, hectares, day_first_sighting, obs_37_norm) %>%
  print(n = 50)
```

```
## # A tibble: 25 x 4
##     year hectares day_first_sighting obs_37_norm
##    <dbl>    <dbl>              <dbl>       <dbl>
##  1  1994     7.81                260       0.931
##  2  1995    12.6                 249       0.863
##  3  1996    18.2                 221       0.826
```

```
##  4  1997     5.77                86       0.843
##  5  1998     5.56                50       0.688
##  6  1999     9.05                 7       0.672
##  7  2000     2.83               249       0.952
##  8  2001     9.35                NA       0.958
##  9  2002     7.54               272       0.924
## 10  2003    11.5                267       0.943
## 11  2004     2.19               116       0.908
## 12  2005     5.92               102       0.933
## 13  2006     6.87               179       0.933
## 14  2007     4.61                95       0.883
## 15  2008     5.06                 1       0.872
## 16  2009     1.92                42       0.826
## 17  2010     4.02               159       0.88
## 18  2011     2.89                60       0.785
## 19  2012     1.19                15       0.766
## 20  2013     0.67                18       0.550
## 21  2014     1.13                54       0.528
## 22  2015     4.01                 7       0.466
## 23  2016     2.91                 7       0.376
## 24  2017     2.48                 1       0.535
## 25  2018     6.05                 9       0.592
```

Feature 3. Number of sightings during the active season (March-October)

Adding a feature that is the number of sightings during the active season divided by/normalied by the total number of sightings for a year. Also going to leave on the total number of observations by year.

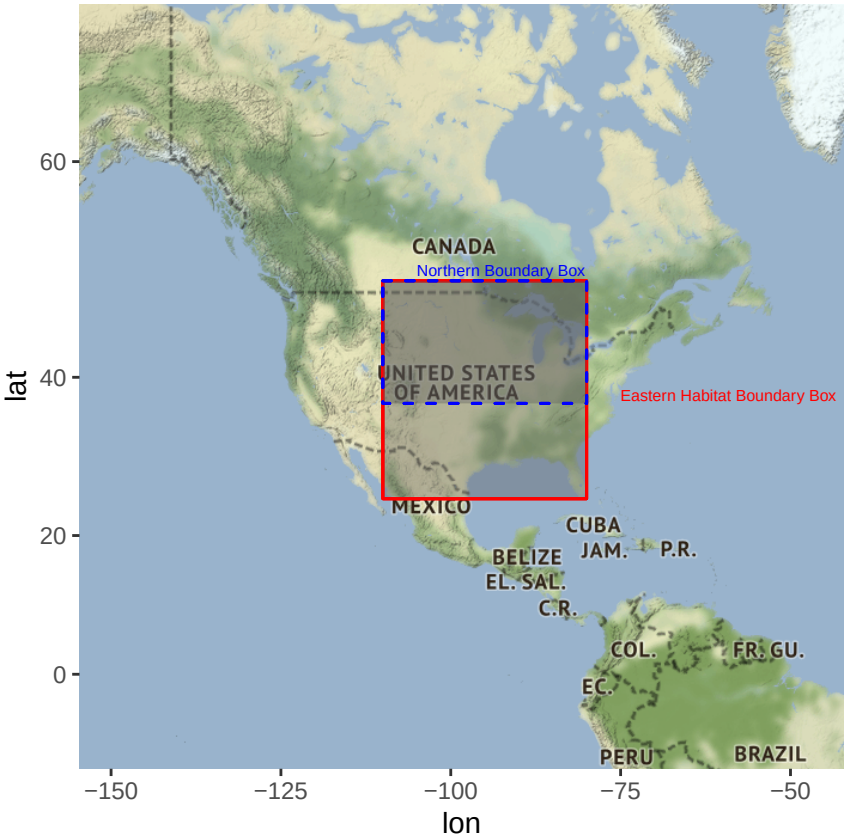
```
monarch_ml_df = monarch %>%
  mutate(year = year(date),
         month = month(date)) %>%
  filter(month %in% 3:10) %>%
  group_by(year) %>%
  summarize(active_months_obs = n()) %>%
  right_join(monarch_ml_df, by = "year") %>%
  left_join(monarch %>%
              mutate(year = year(date)) %>%
              group_by(year) %>%
              summarize(n_obs_total = n()), by = "year") %>%
  mutate(active_months_obs_norm = active_months_obs/n_obs_total) %>%
  dplyr::select(year, hectares, day_first_sighting, obs_37_norm, active_months_obs_norm, n_obs_total) %
  print(n = 50)
```

```
## # A tibble: 25 x 6
##     year hectares day_first_sight~ obs_37_norm active_months_o~ n_obs_total
##    <dbl>    <dbl>            <dbl>       <dbl>            <dbl>       <int>
##  1  1994     7.81              260       0.931            0.982        1246
##  2  1995    12.6               249       0.863            0.969        2091
##  3  1996    18.2               221       0.826            0.957        3833
##  4  1997     5.77               86       0.843            0.981        7367
##  5  1998     5.56               50       0.688            0.931         202
##  6  1999     9.05                7       0.672            0.851         375
##  7  2000     2.83              249       0.952            0.984         249
##  8  2001     9.35               NA       0.958            0.969         262
```

```
##  9  2002     7.54          272        0.924          0.996          236
## 10  2003    11.5           267        0.943          0.980          405
## 11  2004     2.19          116        0.908          0.966          293
## 12  2005     5.92          102        0.933          0.988          416
## 13  2006     6.87          179        0.933          0.978          668
## 14  2007     4.61           95        0.883          0.941          563
## 15  2008     5.06            1        0.872          0.958          569
## 16  2009     1.92           42        0.826          0.915          505
## 17  2010     4.02          159        0.88           0.962          550
## 18  2011     2.89           60        0.785          0.902          592
## 19  2012     1.19           15        0.766          0.902          843
## 20  2013     0.67           18        0.550          0.790          827
## 21  2014     1.13           54        0.528          0.800         1984
## 22  2015     4.01            7        0.466          0.784         3711
## 23  2016     2.91            7        0.376          0.808         8045
## 24  2017     2.48            1        0.535          0.866        15303
## 25  2018     6.05            9        0.592          0.885        28719
```

Features 4-5. Environmental Variables

Adding a bunch of features. The idea here is to generate summaries of bioclim variables (of which there are 19) for two geographic regions. The first is the whole eastern chunk, and the second is the northern range, or the part of range that generates migratory morphs, boundary boxes shown below.



A lot of prism environmental data wrangling that I'm not going to show here, but the big thing is averaging bioclim cells for each area for each year. Also, I added the previous year's count (under the assumption that if you know something about last year's numbers, it might be a good predictor for this year's numbers).

So let's take a look at the final version of the 'real' data.

```
glimpse(monarch_ml_df)
```

```
## Observations: 25
## Variables: 45
## $ year                   <dbl> 1994, 1995, 1996, 1997, 1998, 1999, 200...
## $ hectares               <dbl> 7.81, 12.61, 18.19, 5.77, 5.56, 9.05, 2...
## $ day_first_sighting     <dbl> 260, 249, 221, 86, 50, 7, 249, NA, 272,...
## $ obs_37_norm            <dbl> 0.9309791, 0.8632233, 0.8259849, 0.8430...
## $ active_months_obs_norm <dbl> 0.9815409, 0.9689144, 0.9569528, 0.9805...
## $ n_obs_total            <dbl> 1246, 2091, 3833, 7367, 202, 375, 249, ...
## $ bio_1_whole_range      <dbl> 11.90913, 11.68990, 11.03561, 11.39304,...
## $ bio_2_whole_range      <dbl> 13.27985, 13.08487, 13.21028, 12.74462,...
## $ bio_3_whole_range      <dbl> 34.11212, 34.88401, 34.18034, 33.43231,...
## $ bio_4_whole_range      <dbl> 939.3612, 913.3090, 963.6443, 918.1678,...
## $ bio_5_whole_range      <dbl> 30.48050, 31.70046, 30.55344, 30.56713,...
## $ bio_6_whole_range      <dbl> -9.287580, -6.423049, -8.948854, -8.328...
## $ bio_7_whole_range      <dbl> 39.76808, 38.12351, 39.50230, 38.89587,...
## $ bio_8_whole_range      <dbl> 16.88935, 16.63627, 16.84232, 16.86849,...
## $ bio_9_whole_range      <dbl> 5.548703, 5.732850, 5.125473, 6.756880,...
## $ bio_10_whole_range     <dbl> 22.85669, 23.19043, 22.71369, 22.52431,...
## $ bio_11_whole_range     <dbl> -0.33281242, 0.80273859, -0.30410432, 0...
## $ bio_12_whole_range     <dbl> 807.9618, 814.2814, 810.6394, 834.4942,...
## $ bio_13_whole_range     <dbl> 158.5681, 167.4264, 157.3061, 152.0553,...
## $ bio_14_whole_range     <dbl> 16.977291, 15.237292, 14.683368, 17.506...
## $ bio_15_whole_range     <dbl> 69.32039, 74.74598, 72.40662, 64.79346,...
## $ bio_16_whole_range     <dbl> 326.5077, 357.2198, 342.4300, 335.4309,...
## $ bio_17_whole_range     <dbl> 98.83205, 87.28014, 92.61326, 102.96891...
## $ bio_18_whole_range     <dbl> 254.8641, 245.8748, 252.5326, 240.8461,...
## $ bio_19_whole_range     <dbl> 143.8755, 127.3220, 128.0255, 175.4782,...
## $ bio_1_nrange           <dbl> 8.286128, 7.997299, 7.003093, 7.857129,...
## $ bio_2_nrange           <dbl> 13.16633, 12.65199, 12.74633, 12.52554,...
## $ bio_3_nrange           <dbl> 30.33352, 30.72336, 29.90707, 29.66559,...
## $ bio_4_nrange           <dbl> 1085.8600, 1035.3166, 1107.7406, 1031.0...
## $ bio_5_nrange           <dbl> 28.59613, 30.03058, 28.68773, 28.56437,...
## $ bio_6_nrange           <dbl> -15.226275, -11.318437, -14.432327, -13...
## $ bio_7_nrange           <dbl> 43.82241, 41.34901, 43.12006, 42.54585,...
## $ bio_8_nrange           <dbl> 15.68009, 14.32227, 14.57255, 16.45307,...
## $ bio_9_nrange           <dbl> -1.2172250, -1.2893525, -0.2403663, -0....
## $ bio_10_nrange          <dbl> 20.58555, 21.26873, 20.50640, 20.42132,...
## $ bio_11_nrange          <dbl> -6.003674, -4.207374, -6.142551, -4.558...
## $ bio_12_nrange          <dbl> 636.1724, 708.6670, 720.3555, 651.8420,...
## $ bio_13_nrange          <dbl> 134.9615, 161.5419, 146.3913, 127.7622,...
## $ bio_14_nrange          <dbl> 10.557760, 11.205171, 12.080277, 12.584...
## $ bio_15_nrange          <dbl> 73.78919, 79.01109, 71.06673, 70.19698,...
## $ bio_16_nrange          <dbl> 273.6013, 333.9726, 309.5646, 277.8881,...
## $ bio_17_nange           <dbl> 63.52576, 62.17851, 76.45793, 73.77757,...
## $ bio_18_nrange          <dbl> 228.0936, 229.6106, 217.2065, 241.2476,...
## $ bio_19_nrange          <dbl> 87.56265, 77.42858, 102.49749, 105.2711...
## $ hectare_prev_year      <dbl> NA, 7.81, 12.61, 18.19, 5.77, 5.56, 9.0...
```

So, we've got a really big feature set (43 variables!). For information on what each bioclim variable represents, check out this link.

Synthetic data generation

Here, I use the `synthpop` package to generate 25k observations based on the original data set. `synthpop` uses classification and regression trees (though you can use other predictive algorithms). Below, I'll demonstrate the code, and also look at some comparisons between the real data dn the synthetic data.

```
library(synthpop)

#reading in the 'real' data
monarch_data =read_csv("./data/monarch_data_real.csv")

#Generating a list for all the variables indicating that we want to use density smoothing
smooth_list = as.list(rep("density", 45))
names(smooth_list) = names(monarch_data)

#Generating synthetic data
syn_monarch = synthpop::syn(monarch_data, k = 25000, smoothing = smooth_list)
```

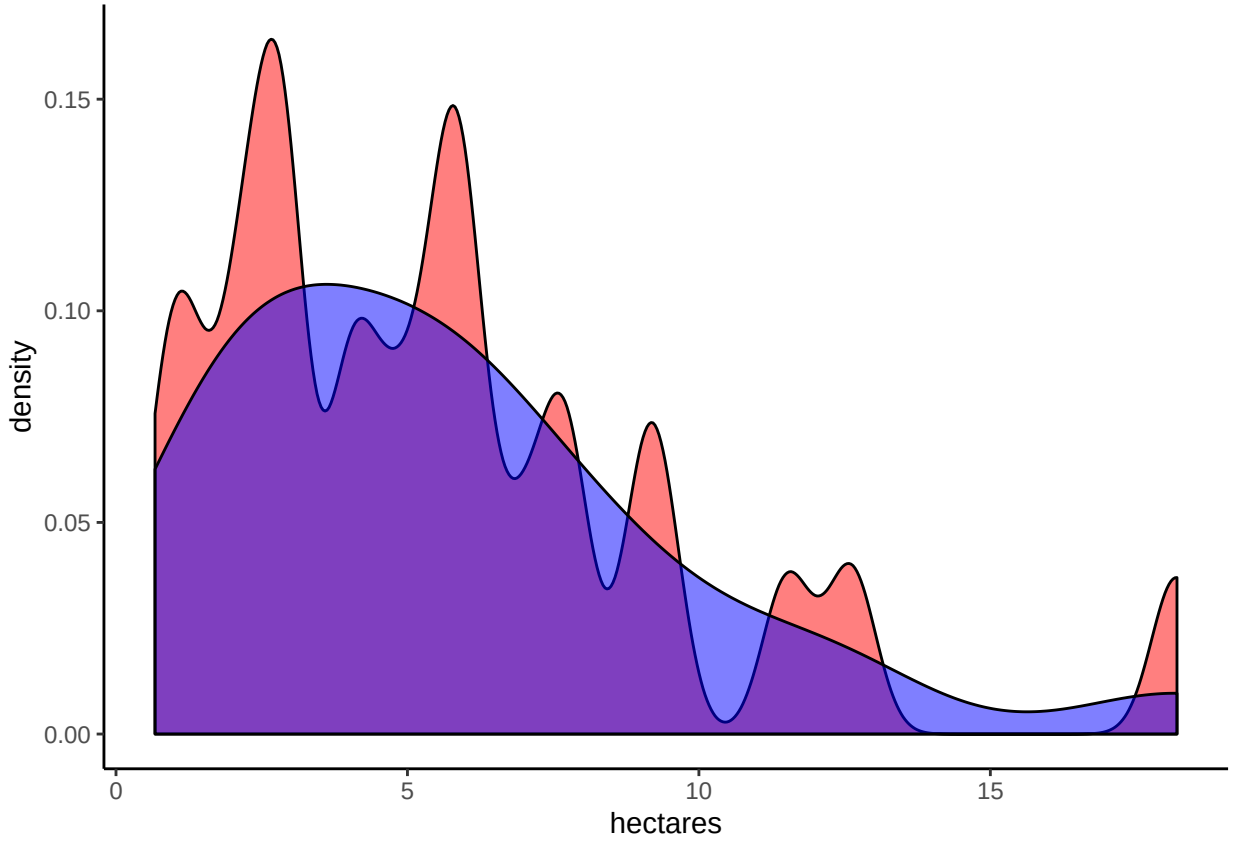
```
## Sample(s) of size 25000 will be generated from original data of size 25.
##
## Synthesis
## -----------
##  year hectares day_first_sighting obs_37_norm active_months_obs_norm n_obs_total bio_1_whole_range bi
##  bio_5_whole_range bio_6_whole_range bio_7_whole_range bio_8_whole_range bio_9_whole_range bio_10_wh
##  bio_15_whole_range bio_16_whole_range bio_17_whole_range bio_18_whole_range bio_19_whole_range bio_
##  bio_6_nrange bio_7_nrange bio_8_nrange bio_9_nrange bio_10_nrange bio_11_nrange bio_12_nrange bio_13
##  bio_16_nrange bio_17_nange bio_18_nrange bio_19_nrange hectare_prev_year
```

```
glimpse(syn_monarch$syn)
```

```
## Observations: 25,000
## Variables: 45
## $ year                   <dbl> 2003, 2009, 2008, 2002, 1999, 2011, 201...
## $ hectares               <dbl> 12.6205086, 0.6835774, 6.8688495, 5.791...
## $ day_first_sighting     <dbl> 260, 1, 54, 179, 249, 9, 9, 1, 18, 159,...
## $ obs_37_norm            <dbl> 0.9436217, 0.6720552, 0.9579208, 0.8804...
## $ active_months_obs_norm <dbl> 0.9821703, 0.8663978, 0.9578724, 0.9579...
## $ n_obs_total            <dbl> 297, 547, 827, 238, 297, 829, 543, 1984...
## $ bio_1_whole_range      <dbl> 11.90299, 13.35506, 11.39061, 12.75165,...
## $ bio_2_whole_range      <dbl> 13.17162, 13.84739, 12.89622, 13.29558,...
## $ bio_3_whole_range      <dbl> 34.26495, 36.29856, 32.97901, 36.52387,...
## $ bio_4_whole_range      <dbl> 939.3973, 877.4746, 979.8729, 913.2669,...
## $ bio_5_whole_range      <dbl> 31.60195, 31.70709, 31.65346, 31.32308,...
## $ bio_6_whole_range      <dbl> -9.283842, -6.427118, -10.119984, -6.14...
## $ bio_7_whole_range      <dbl> 39.77712, 38.12518, 39.23826, 38.15555,...
## $ bio_8_whole_range      <dbl> 16.36003, 16.35189, 18.67064, 17.93033,...
## $ bio_9_whole_range      <dbl> 5.124645, 6.879749, 5.133792, 6.882429,...
## $ bio_10_whole_range     <dbl> 23.07506, 23.80393, 23.58844, 23.65694,...
## $ bio_11_whole_range     <dbl> -1.53574871, 0.80258663, 0.11921283, 1....
## $ bio_12_whole_range     <dbl> 834.8954, 771.1961, 906.9930, 805.9708,...
## $ bio_13_whole_range     <dbl> 158.7002, 158.8645, 179.0031, 158.5361,...
## $ bio_14_whole_range     <dbl> 10.926046, 12.841135, 10.921268, 11.385...
## $ bio_15_whole_range     <dbl> 77.80523, 77.80847, 72.74608, 75.57488,...
## $ bio_16_whole_range     <dbl> 305.8017, 328.8095, 375.1671, 305.8420,...
## $ bio_17_whole_range     <dbl> 90.37981, 70.29182, 70.33735, 84.24732,...
```

```
## $ bio_18_whole_range     <dbl> 240.9622, 218.1543, 275.3787, 208.1027,...
## $ bio_19_whole_range     <dbl> 128.0754, 143.8449, 144.0382, 145.0818,...
## $ bio_1_nrange           <dbl> 8.361840, 9.511409, 7.005360, 8.846852,...
## $ bio_2_nrange           <dbl> 13.02762, 13.79615, 12.64215, 13.26598,...
## $ bio_3_nrange           <dbl> 30.64237, 30.72861, 30.86298, 30.72497,...
## $ bio_4_nrange           <dbl> 1079.4549, 1036.2265, 1059.0995, 942.45...
## $ bio_5_nrange           <dbl> 30.06397, 29.50354, 29.49309, 29.84492,...
## $ bio_6_nrange           <dbl> -12.868444, -12.020133, -12.745637, -12...
## $ bio_7_nrange           <dbl> 43.82112, 41.54558, 42.30591, 40.89967,...
## $ bio_8_nrange           <dbl> 16.28704, 15.00883, 15.32571, 14.32731,...
## $ bio_9_nrange           <dbl> -0.6402462, -0.5589204, 0.7210804, -0.6...
## $ bio_10_nrange          <dbl> 21.61088, 21.11888, 21.86425, 21.78309,...
## $ bio_11_nrange          <dbl> -5.310882, -3.987473, -4.758633, -2.892...
## $ bio_12_nrange          <dbl> 560.3515, 646.8567, 777.2848, 655.8160,...
## $ bio_13_nrange          <dbl> 138.0351, 139.7812, 172.1431, 137.9596,...
## $ bio_14_nrange          <dbl> 10.559440, 11.202647, 12.087530, 10.559...
## $ bio_15_nrange          <dbl> 73.79452, 71.05263, 73.85673, 79.02136,...
## $ bio_16_nrange          <dbl> 307.9949, 314.0783, 363.7336, 312.0911,...
## $ bio_17_nange           <dbl> 53.61148, 69.43739, 73.85538, 85.38710,...
## $ bio_18_nrange          <dbl> 248.6372, 229.7460, 217.3158, 248.7310,...
## $ bio_19_nrange          <dbl> 87.60077, 77.47612, 95.21811, 98.05713,...
## $ hectare_prev_year      <dbl> 1.9211141, 5.0433581, 4.0228482, 5.9361...
```

```r
#Plotting
ggplot(data = syn_monarch$syn, aes(x = hectares)) +
  geom_density(fill = "red", alpha = 0.5) +
  geom_density(data = monarch_data, aes(x = hectares), fill = "blue", alpha = 0.5) +
  theme_classic()
```



So this is interesting - it looks like the synthetic data generation is matching the original data set very closely but magnifying small-scale structure. In the above graph blue is the original kernel density estimate and red is the synthetic data. Need to look at ways to smooth the curve so it more closely matches the original. Perhaps another algorithm that isn't quite as good at predicition would do the trick.

Preprocessing the data prior to algorithm building

We're going to use the `recipes` package to build a pre-processing recipe that we'll then apply to the training (synthetic) and test (real) sets before we start algorithm building. The main preprocessing steps we want to accomplish are:
1. **k-nn imputation to fill in any missing values**
2. **Center and scale the data**
3. **Remove any variables where there is zero or near-zero variance**
4. **Also tried some PCA on bioclim variables, but resulted in large drops in predictive power...ended up removing this**

Algorithm building

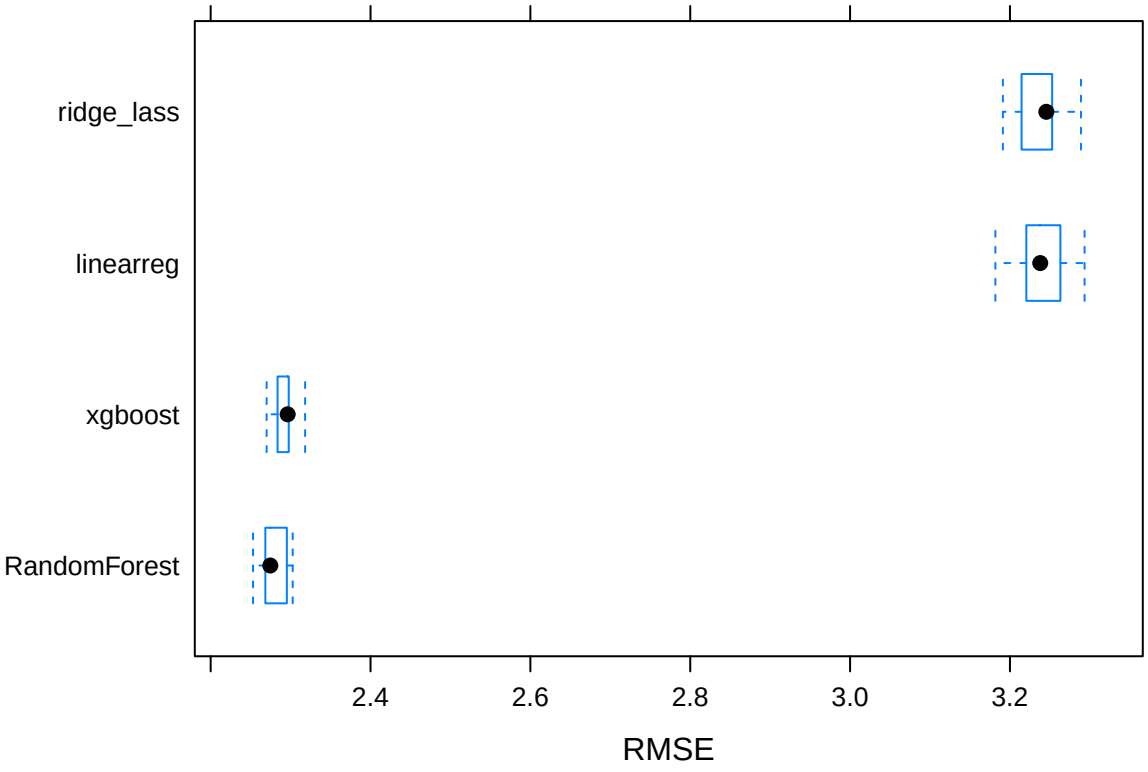
I built and tuned four algorithms with 5-fold cross validation with 5 repeats:
1. **Linear regression**
2. **Random Forest**
3. **Extreme Gradient Boosting**
4. **Ridge and Lasso Regression**

```
#Let's compare models

#Training data evaluation
results <- resamples(list(RandomForest=rf_mod, linearreg=lm_mod, xgboost = xgboost_mod, ridge_lass = ri

# summarize the distributions
summary(results)
```

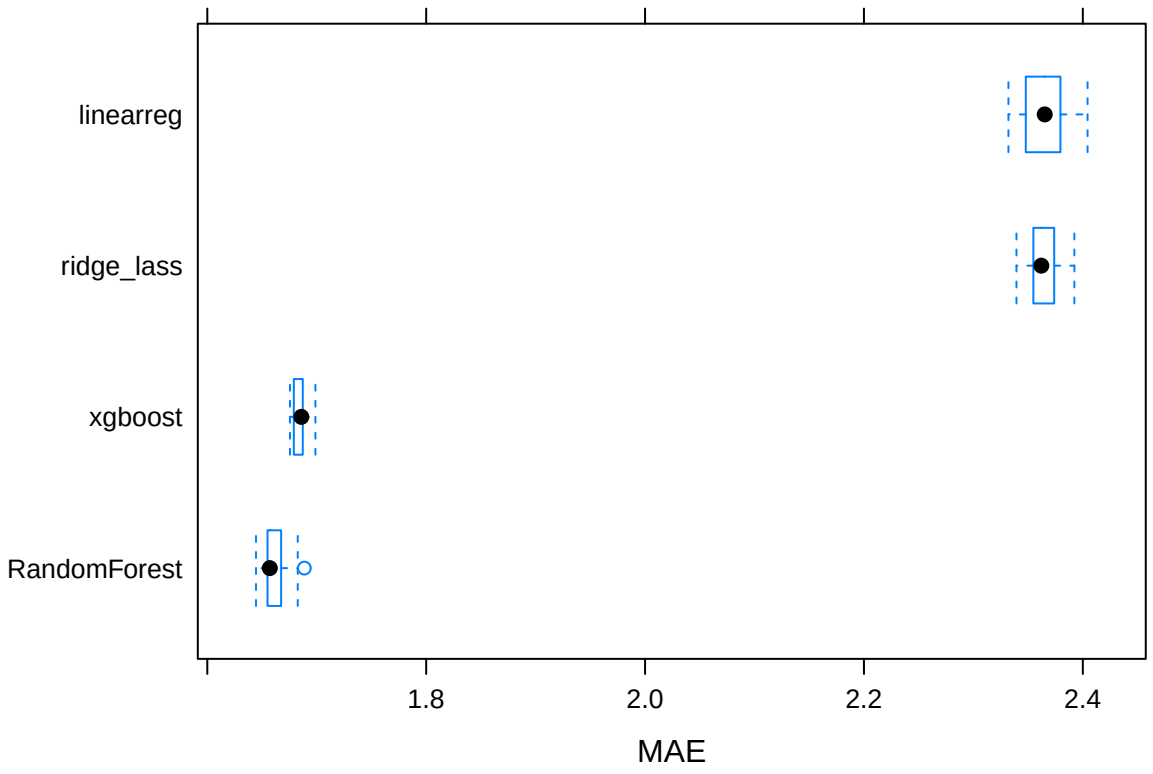
```
##
## Call:
## summary.resamples(object = results)
##
## Models: RandomForest, linearreg, xgboost, ridge_lass
## Number of resamples: 9
##
## MAE
##                   Min.  1st Qu.   Median     Mean  3rd Qu.     Max. NA's
## RandomForest 1.644480 1.655004 1.657120 1.662680 1.667468 1.688675    0
## linearreg    2.332024 2.347880 2.365171 2.366033 2.379547 2.404310    0
## xgboost      1.675524 1.679049 1.685908 1.684636 1.687221 1.698794    0
## ridge_lass   2.339345 2.354797 2.362054 2.363989 2.373838 2.392277    0
##
## RMSE
##                   Min.  1st Qu.   Median     Mean  3rd Qu.     Max. NA's
## RandomForest 2.252960 2.268428 2.274635 2.277729 2.295307 2.302664    0
## linearreg    3.181752 3.220479 3.237910 3.241205 3.263079 3.293360    0
## xgboost      2.270002 2.283686 2.296312 2.294362 2.297662 2.318161    0
## ridge_lass   3.191147 3.214555 3.245498 3.239966 3.252830 3.288927    0
##
```

```
## Rsquared
##                    Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## RandomForest 0.6699582 0.6722862 0.6736747 0.6741725 0.6748555 0.6794230
## linearreg    0.3313659 0.3362955 0.3387148 0.3397357 0.3430991 0.3488801
## xgboost      0.6626432 0.6692941 0.6702330 0.6693079 0.6708356 0.6722146
## ridge_lass   0.3353143 0.3371205 0.3405813 0.3402218 0.3434402 0.3472174
##              NA's
## RandomForest    0
## linearreg       0
## xgboost         0
## ridge_lass      0
```

```
# boxplot of results
bwplot(results, metric="RMSE")
```




```
bwplot(results, metric="MAE")
```




```
#Test Data evaluation
monarch_test_data = read_csv("./data/monarch_test_data.csv")
```

```
## Parsed with column specification:
## cols(
##   .default = col_double()
## )
```

```
## See spec(...) for full column specifications.
```

```
#Making predictions
rf_mod_fit = predict(rf_mod, monarch_test_data)
xgboost_mod_fit = predict(xgboost_mod, monarch_test_data)
lm_mod_fit = predict(lm_mod, monarch_test_data)
ridge_lasso_fit = predict(ridge_lasso_mod, monarch_test_data)

#Post resample on test data
postResample(pred = rf_mod_fit, obs = monarch_test_data$hectares)
```

```
##      RMSE  Rsquared       MAE
## 2.1832356 0.7102117 1.6071563
```

```
postResample(pred = xgboost_mod_fit, obs = monarch_test_data$hectares)
```

```
##      RMSE  Rsquared       MAE
## 2.1591925 0.7146667 1.6525019
```

```
postResample(pred = lm_mod_fit, obs = monarch_test_data$hectares)
```

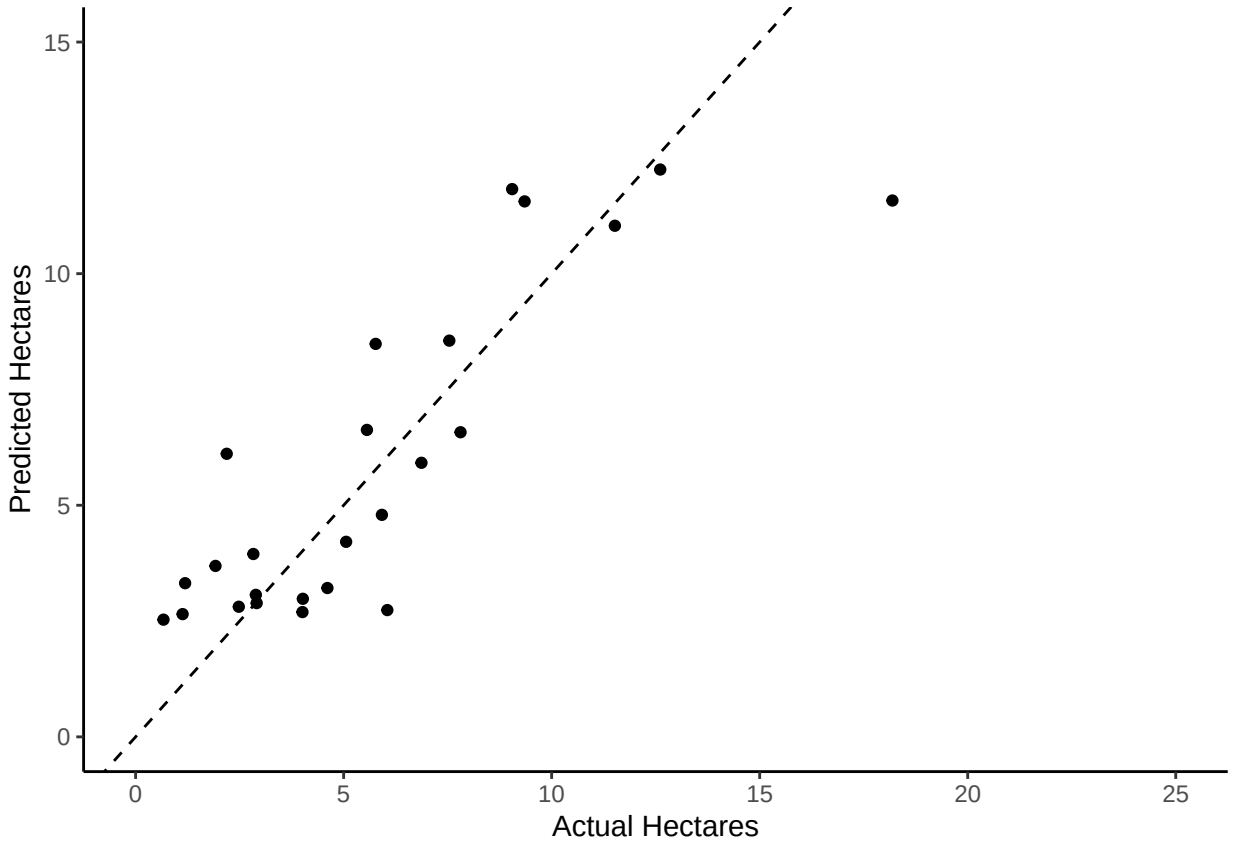
```
##      RMSE  Rsquared       MAE
## 3.1806488 0.3887398 2.3772050
```

```
postResample(pred = ridge_lasso_fit, obs = monarch_test_data$hectares)
```

```
##      RMSE  Rsquared       MAE
## 3.1788589 0.3911055 2.3759039
```

So, overall these are pretty decent as a first pass. The best model, the extreme gradient boosting model is explaining **over 71%** of the variance in wintering numbers and has a mean absolute errors of about 1.65 hectares.
We can also visualize the predictions against the real results for a bit more insight.

```
#binding on predicitions
monarch_test_data$pred = xgboost_mod_fit

ggplot(monarch_test_data, aes(x = hectares, y = pred)) +
  geom_point() +
  geom_abline(slope = 1, intercept = 0, lty = 2) +
  theme_classic() +
  xlab("Actual Hectares") +
  ylab("Predicted Hectares") +
  xlim(c(0,25)) +
  ylim(c(0,15))
```



Overall, this is a pretty good fit - though on average it looks like low values are consistently over predicted and there is one high value that is *WAY* under-predicted. The dashed line above is the 1:1 line - representing perfect prediction.

We can also try and try and glimpse inside the black box by looking at variable importance.

```
varImp(xgboost_mod)
```

```
## xgbTree variable importance
##
##   only 20 most important variables shown (out of 43)
##
##                          Overall
## bio_8_whole_range      100.0000
## bio_14_whole_range      63.3526
## bio_19_whole_range      57.5233
## n_obs_total             33.0986
## day_first_sighting      16.5406
## active_months_obs_norm   5.7783
## bio_5_whole_range        5.6765
## obs_37_norm              2.3336
## bio_2_whole_range        1.5265
## bio_10_whole_range       1.3207
## bio_5_nrange             0.7249
## bio_12_whole_range       0.2766
## bio_8_nrange             0.2508
## bio_15_whole_range       0.2477
## bio_17_whole_range       0.2340
## bio_9_whole_range        0.1923
## hectare_prev_year        0.1902
## bio_14_nrange            0.1756
## bio_11_whole_range       0.1654
## bio_7_whole_range        0.1114
```

This is super interesting. Bioclim 8, 14, 19 are the most importance variables, which are mean temperature of the wettest quarter, precipitation of the driest month and precipitation of the coldest quarter, respectively. Interesting to think about why these might pop out and how they're related to monarch biology.

Other important factors include total number of observations, and day of first sighting. Number of observations is problematic, because it's not scaled for effort - but may be correlated with a strong downward trend in population numbers (i.e. more recent records always have more total observations because of high effort, but more recent records also have lower numbers because of declines). Need to figure out a good way to control for this in this context.

Future Directions

This is a first pass. I think we can improve the predictive power of the model in a number of ways.

1. **Larger synthetic data set.** These were built/trained with 25k records - we can definitely increase this. Something like 100k or 500k would be great, but it's going to require substantial computing power on Cyverse.
2. **Adaptive Tuning.** The `caret` package supports adaptive tuning where once we have a set of model parameters that are producing good results we can narrow the search grid to fine tune these parameters.
3. **Ensemble models.** I don't show it hear, but the models have fairly low correlations among each other, making them a good fit for ensemble methods. This has worked in a past project I did to increase accuracy.
4. **Fine tuning the input variables.**. There are probably other variables that we could think of that are important and we might be able to add. Adding more important biological features would probably not be a bad idea.