# Materials and methods

Emilio Berti

This is a summary of the steps taken so far to achieve a dataset that can be used to generate fluxes. Four main steps were performed so far:

1. Harmonize list of species names following the GBIF taxonomic backbone.
2. Impute missing body masses and combine body mass datasets with tetradensity clipped to Europe.
3. Extract environmental variables for tetradensity coordinates.
4. Model species density following Santini et al. (2018).

From this, it is still needed to:

1. Prepare a dataset that has, at the pixel level, all information needed to run fluxweb.

## Source datasets

**The list of European vertebrates** comes from the *FutureWeb* project. Species names were standardized using ITIS (even in couple of cases this was dubious to me).

**Body mass** data comes from:

- EuropAmphib for amphibians (not sure where this is from).
- EltonTraits for birds.
- EltonTraits for mammals.
- Slavenko et al. () for reptiles.

**Environmental data** (NPP, temperature, precipitation) comes from CHELSAE (not sure what to reference here, as I think these were custom generated).

**Species density** comes from the TetraDENSITY dataset (Santini et al., 2018).

## Taxonomic harmonization

Initially, we decided to use another dataset for reptiles, but I belive this to be incomplete and with some issues (I think it become incomplete after it was wrangled and I cannot load *.accdb* from Linux). I used the four above as they are a quite standard source for vertebrate traits in macroecology.

I combined all species names from all datasets into a unique list that was harmonized using *rgbif*. In particular, I did a first search to find the name and, in the case this was a non-accepted taxonomic name (e.g. a synonym), I performed a second search to find the accepted taxonomic name. The script to run for this is `harmonize-taxonomy.R`, with harmonized taxonomic backbone saved as **data/raw/backbone.csv**.

# Combine datasets

I load TetraDensity and the body mass datasets and assign the new harmonized taxonomic backbone (*data/raw/backbone.csv*) to species names. I then combined all body mass datasets together. To and the resulting dataframe with TetraDensity, which was previously clipped to retain only European data points. The result is a shapefile with species harmonized taxonomies, body mass, and density.

I imputed all missing body masses using a multi-imputation (chained equations) method and a predictive mean matching algorithm. Missing body mass values were imputed as the Bayesian stochastic regression using taxonomic family as predictor. Imputation performed moderately well, with all chains converging and with a total efficiency of .999 ($\gamma = 0.013$).

After imputations, I added body masses and if they were imputed to the taxonomic backbone, TetraDENSITY shapefile, and FutureWeb trait table. The script to run for this is `combine_databases.R`, with files saved as: **data/interim/backbone-masses.csv**, **data/interim/tetradensity-masses.shp**, and **data/interim/futureweb-masses.csv**.

# Extract environmental variables

This is quite straightforward and I simply extracted from the environmental rasters the values at the TetraDENSITY locations. Environmental layers were: average annual temperature, precipitation seasonality (PCV), and net primary productiviy (NPP). Importantly, I aggregated rasters to have 1 arc-degree resolution, as this was the resolution more appropriate to model density data (see also Santini et al., 2018). During aggregatation, each new cell had value equal to the median value in the region for PCV and temperature and the sum of the NPP values. As these operation are all quite fast to perform, I did not save the new layers, but used them directly in the next step.

# Model species density

For each class (Amphibia, Aves, Mammalia, Reptilia), I subsetted the data to retain the specific class and initialized a full model following Santini et al. (2018). An important change from the Santini et al. (2018) model is that I included only taxonomic order and not other taxonomic information in the model. As Santini et al. (2018) aimed to explained environmental drivers of species density, taxonomic information was included as random effects, which explained most of the variability in the data. However, as we are interested here in predicting species density, including taxonomic order as random effect would bias our predictions; taxonomic rank was thus included as a fixed effect. Moreover, I did not include other random effects (e.g. pixel identity), which were included in Santini et al. (2018) to capture the variability in the data.

Following Santini et al. (2018) For each class, the full model was:

$$Density \sim log_{10}(Mass) + log_{10}(Mass)^2 + log_{10}(NPP) + log_{10}(NPP)^2 + PCV + PCV^2 + temperature + Order$$

For each model, we performed a model selection to retain only important predictors. This is summarized in `check_models.R`. I retained all predictors that were present at least in one submodel within a $\Delta AIC < 2$. As all predictors were in at least one of such submodels, densities were predicted using the full model for all classes.

When trying to save the models, I encountered a weird thing, i.e. they would take a lot of space (~80Gb). I'm not sure why this happens, but, as the models are very fast to run, I preferred to not save them. Instead, the script `make_density_models.R` can be run to obtain the individual models, contained in a named list:

```
names(lm_density)
```

```
## [1] "Amphibia" "Aves"     "Mammalia" "Reptilia"
```

```
lm_density["Amphibia"]
```

```
## $Amphibia
##
## Call:
## lm(formula = log10(Density) ~ log10(Mass) + I(log10(Mass)^2) +
##     log10(NPP) + I(log10(NPP)^2) + PCV + I(PCV^2) + Order + Temp,
##     data = d %>% filter(Class == "Amphibia"))
##
## Coefficients:
##      (Intercept)       log10(Mass)  I(log10(Mass)^2)        log10(NPP)
##       -3.424e+01        -9.436e-01         5.567e-01         9.471e+00
##  I(log10(NPP)^2)               PCV          I(PCV^2)     OrderCaudata
##       -6.573e-01         9.375e-04        -1.227e-07         2.175e+00
##             Temp
##       -9.754e-03
```

Plots of model residuals (for checking assupmtions) and quality of predictions can be found in **docs/plots/.pdf**

## Predict densities for FutureWeb

As FutureWeb has more taxonomic orders than TetraDENSITY, it is not possible to predict densities of all species without imputing missing values or running a different modeling framework, e.g. taxonomic order as random intercept. A solution is to drop missing values. I explored how many interactions would be removed by dropping rows for which the taxonomic order of prey and predator do not appear in TetraDENSITY:

```
read_csv("../docs/proportion-of-interactions-modelled.csv") %>%
  knitr::kable()
```

```
## Rows: 15 Columns: 5
```

```
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (2): Class.source, Class.target
## dbl (3): complete, missing, complete fraction
```

```
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

| Class.source | Class.target | complete | missing | complete fraction |
|---|---|---:|---:|---:|
| Aves | Mammalia | 7546 | 530 | 0.9343735 |
| Aves | Reptilia | 5383 | 368 | 0.9360111 |
| Aves | Amphibia | 4154 | 205 | 0.9529709 |
| Mammalia | Aves | 2990 | 98 | 0.9682642 |
| Aves | Aves | 7640 | 71 | 0.9907924 |
| Mammalia | Mammalia | 3468 | 48 | 0.9863481 |
| Reptilia | NA | 0 | 28 | 0.0000000 |
| Aves | NA | 0 | 12 | 0.0000000 |
| Amphibia | Amphibia | 4 | 0 | 1.0000000 |
| Mammalia | Amphibia | 1329 | 0 | 1.0000000 |
| Mammalia | Reptilia | 2733 | 0 | 1.0000000 |
| Reptilia | Amphibia | 1341 | 0 | 1.0000000 |

| Class.source | Class.target | complete | missing | complete fraction |
|---|---|---:|---:|---:|
| Reptilia | Aves | 6322 | 0 | 1.0000000 |
| Reptilia | Mammalia | 8965 | 0 | 1.0000000 |
| Reptilia | Reptilia | 7929 | 0 | 1.0000000 |

The columns show the class of the prey (*.source*) and predator (*.target*), with completed being the number of rows with complete information, and missing the number of rows without complete information, with *complete fraction* being the ratio of the two.