Applied Ecological Statistics - ECO 636

Lab 7: Multimodel Inference

1 Introduction

This lab explores the application of information-theoretic approaches to model selection and multi-model inference. Specifically, it provides you with tools for dealing with multiple competing hypotheses, and multiple candidate models. This will allow you to evaluate the support for a particular model, assess variable importance, and make predictions using all or subsets of the models while accounting for uncertainty in the model selection process. Here, you will learn the ins and outs of model selection, model weights, and model averaging. You will practice this by working through the process of developing and fitting a set of candidate models, using AIC and its variants (e.g., AICc) to quantitatively evaluate the support for these models, and use these model weights to make model-averaged predictions and generate model-averaged parameter estimates. Today is a little bit of a decviation from our normal labs, in that this one is a stand-alone lab without a lecture counterpart!

2 Formulating Competing Hypotheses

The first step in any analytical endeavor is to specify the hypotheses you wish to evaluate. While in some cases we may have a single hypothesis to test (e.g., *does mean home range size differ between males and females?*), in the real world, you are much more likely to have several competing hypotheses. For example, you may be interested in understanding the influence of surrounding landscape on abundance of a particular species. Abundance may be influenced by a range of factors, including natural habitat features (e.g., riparian habitat, early succession forest), or features of landscape disturbance (e.g., roads, urban development). You may be interested in the effect of each feature on its own, or in combination. Each of these hypotheses can be represented by a statistical model which allows formal "testing" of each hypothesis. In a classical hypothesis framework, we would generally test if our model parameter estimates (i.e., $\hat{\beta}$) are significantly different from zero using p-values and a specified

alpha level (typically $\alpha = 0.05$).

Quiz yourself: Why do you think $\alpha = 0.05$ is the *assumed* value for hypothesis testing? Is there any argument you can make for or against it?

No matter what α value is used, biologists have recognized the problems with null hypothesis testing, particularly when trying to understand multiple competing models and make informed inference. Two of the big trouble spots are: 1) how to formally evaluate the degree of support for each model in a set of candidate models, and 2) how to incorporate information from more than one model in ecological inference.

On top of that, null hypothesis test based inference uses a potentially arbitrary α value where parameters and models whose p-values *just* exceeds alpha (e.g., $p = 0.052$) are ignored. Ideally, we would like to simultaneously compare multiple competing hypotheses and quantify the degree of support for each. Information-theoretic approaches to model selection allow us to do exactly this.

Before diving in, stop and think about how multiple competing hypotheses are generated. One approach is to include a relatively small number of models selected *a priori* based on careful ecological consideration of the study system, previous studies, and the research objectives. This means the results may be thought of as either confirming or denying the researcher's *specific* hypotheses. In other contexts, such as exploratory or observational studies, there may be little or no *a priori* ecological rationale to consider one model over another. If there is a good reason to suspect all independent variables (covariates) are biologically relevant, a logical approach might be to consider all possible combinations of those variables. Critics of such an *all-subsets* approaches argue this borders on data-dredging and fosters a lack of critical thinking about the study system, but, others argue that only investigating "expected" hypotheses risks ignoring potentially novel or unanticipated relationships in the data.

Quiz yourself: think about the *positives* and *negatives* to these different approaches. Which do you think is more applicable for *your* research?

It is important to recognize that these two approaches represent different ends of a continuum and the ideal approach for any given study will depend on the study system, the objectives, and the data. In any situation, it is very important to think carefully and critically about the independent variables that are used, and the way in which those variables are combined to create a candidate model set. Once a candidate model set is generated, the next step is to evaluate the support for each model. As always, remember to make sure you follow the modeling framework we've established, and always assess your data's fit to the linear modeling assumptions. In today's lab we will explore these ideas using the familiar *indigo*

snake data and demonstrate multimodel inference in practice.

3 Indigo Snake Example

3.1 Background

This data come from former ECo student Javan Bauder's PhD research examining the effects of landscape composition on the home range size of the federally threatened eastern indigo snake, *Drymarchon couperi*. The indigo snake (EIS) have some of the largest home ranges reported for terrestrial snakes (>1,500-ha in parts of their range). This study was conducted at several sites in central Florida in landscapes ranging from unfragmented natural habitats to high-density urban development. The working hypothesis is that EIS homerange size is influenced by three land cover types, wetland, urban, and terrestrial upland (all non-wetland habitats with minimal human disturbance). The amount of each land cover type within a 1.50-km radius extending from the centroid of each snake's home range was measured. In addition, the sex and number of days each snake was tracked was recorded. Because home range size often varies based on effort, the number of days each snake was tracked was included in the modeling to control for this variation.

Quiz yourself: why might the number of days tracked influence observed home range size?

As a first step, Javan thought through a small number of *a priori* hypotheses about factors influencing EIS home range size to begin generating a list of candidate models. Previous research has shown that EIS home range size is significantly larger in males than in females, so it makes sense to include sex in all competing models. Tracking duration varied among individual snakes, so, number of days tracked should be included as a fixed effect in each model (i.e. as a control). Land cover also may have a strong influence. In fact, previous research suggests that EIS respond negatively to urban development and prefer terrestrial upland and wetland habitats, therefore, a reasonable hypothesis is that EIS home range will be influenced by all land cover types. It is less obvious, however, whether any particular combination of these three variables represents a more or less biologically-reasonable hypothesis.

Quiz yourself: with so few covariates, (sex, duration, upland, wetland, and urban) do you think taking **all** subsets could be thought of as data dredging (i.e., looking for any pattern/relationships without regard to ecology/system)?

We have *a priori* reasons to include sex and tracking days in every model, and since we are primarily interested in evaluating the effects of land cover on EIS home range size there would be no reason not to include all combinations of cover types in the model sets. Whatever approach is taken, the final candidate model set should **aways** be selected with thought and consideration, and the rationale for the choice of covariates and candidate models should be explicitly stated. It's worth noting that the model set may change over the course of the analysis as you deal with issues such as collinearity, variable insufficiency, outliers, etc.

Lets get the data and load some required libraries, specifically `ggplot2` and `HH` for plotting, and `AICcmodavg` and `MuMIn` for doing model selection and multi-model inference. Although `AICcmodavg` and `MuMIn` have many similar functions, `MuMIn` has more versatility but `AICcmodavg` is slightly more user-friendly for some things. It is always good practice to load all packages at the start of your `R` script rather than one at a time throughout (or in the `setup` chunk if you are using `R markdown`).

```
library(AICcmodavg)
library(MuMIn)
library(ggplot2)
library(HH)
source("multiplot.r")
indigos <- read.table(file = "indigos.txt", header = TRUE)
```

The response variable in this example is EIS home range size. Since we used this data in class using a log-transform, we will do the same here today and use an *identity* link in our GLM to meet the model assumptions.

Quiz yourself: what other types of GLM links could you use with the *raw* data to meet model assumptions?

We have five covariates, four continuous (`ndays`, `urban1.50`, `upland1.50`, `wetland1.50`), and one categorical (`sex`). Prior to model fitting, a z-score standardization will be applied our continuous data using the `scale()` function. This puts that data on the same scale making it easier to directly compare our parameter estimates. Finally, storing each fitted model in a named list is convenient for use with some of the functions we will use later.

```r
# Log-transform home range size
indigos$logHR <- log(indigos$hr.size)
# Scale the following columns in the `indigos` data set
indigos$Sndays       <- c(scale(indigos$ndays))
indigos$Surban1.50   <- c(scale(indigos$urban1.50))
indigos$Supland1.50  <- c(scale(indigos$upland1.50))
indigos$Swetland1.50 <- c(scale(indigos$wetland1.50))

# Create a list of 8 models
models <- list(
  "Null" = glm(logHR ~ sex + Sndays, indigos,family="gaussian"),
  "Urb" = glm(logHR ~ sex + Sndays + Surban1.50, indigos, family="gaussian"),
  "Up" = glm(logHR ~ sex + Sndays + Supland1.50, indigos, family="gaussian"),
  "Wet" = glm(logHR ~ sex + Sndays + Swetland1.50, indigos,
              family="gaussian"),
  "Urb+Up" = glm(logHR ~ sex + Sndays + Surban1.50  + Supland1.50, indigos,
                 family="gaussian"),
  "Urb+Wet"= glm(logHR ~ sex + Sndays + Surban1.50  + Swetland1.50, indigos,
                 family="gaussian"),
  "Up+Wet" = glm(logHR ~ sex + Sndays + Supland1.50 + Swetland1.50, indigos,
                 family="gaussian"),
  "Urb+Up+Wet" = glm(logHR ~ sex + Sndays + Surban1.50 + Supland1.50 +
                  Swetland1.50, indigos, family="gaussian"))
```

Wait a minute, there is a model here called the *null* model but it doesnt look like our usual "intercept-only" model. The intercept-only model is often a "straw man" (i.e., a null that is guaranteed to perform poorly). Because sex and the number of tracking days are known to explain variation in home range size, and because we are specifically interested in how land cover (not `sex` or `ndays`) affects EIS home range size, the null model here is the model with *only* sex and number of days. This is commonly referred to as an "informed null" hypothesis.

Okay, now that the models have been fitted, let's move on to formally evaluating the degree of support for each model.

3.2 Model Selection and Comparison

Once you've established a candidate model set, the weight of evidence for each model can be calculated. As we have introduced in class, information criteria (IC), which have their basis in information theory, provide the means to do so. The most popular IC used in ecology is the Akaike Information Criteria (AIC: Akaike, 1973) and today we'll look at it more in depth than we covered in class. But first, a refresh: the formula for AIC is very simple: -2 times the log-likelihood (LL) plus 2 times the number of model parameters.

$$\text{AIC} = \text{-}2LL + 2K$$

AIC is an approximate measure of the distance from your model's estimation and the truth. But, since we don't actually know the *truth*, it uses an estimate of the the "Kullback-Leibler"" (K-L) information lost when a model approximates reality. Reality is an unknown constant and so drops out of the calculations making AIC a measure of *relative* model performance (relative to other models in the 'set'). AIC also includes a penalty term for the number of parameters. Otherwise, the likelihood term (-2LL) would be continually improved by adding parameters which in turn increases the risk of model-over-fitting. AIC thereby works to find a balance between under-fitting a model (which increases its *bias*) and over-fitting a model (which increases its *variance*).

Quiz yourself: can you explain AIC in non-stats language?

The smaller the AIC the "closer" (in KL information space) a model is to reality, so the model with the smallest AIC is considered the 'best' model in the candidate set. However, it is important to remember that AIC is a relative measure of model performance. The AIC-best model may explain very little of the variation in the data or have poor a goodness-of-fit. Therefore, it's always important to make sure that model assumptions are being met and that measures of absolute performance are considered (e.g., R^2, goodness-of-fit tests).

The `AIC()` function can be used to return the AIC of any `R` object for which the log-likelihood can be extracted.

```
# Extract AIC using AIC function for the Urb+Up model
AIC(models[["Urb+Up"]])
```

One frequently used variant of AIC, particularly with ecological data, is *AICc*, often called second-order AIC or small-sample AIC. AICc penalizes more complex models at smaller

sample sizes:

$$\text{AICc} = \text{AIC} + \frac{2K(K+1)}{n-K-1}$$

Because AICc approaches AIC at large ratios of n/K (> 40), some recommend always using AICc.

The `AICcmodavg` package has a function, `aictab()`, that calculates AICc (and other values) for each model in the list of models. In the package `MuMIn`, the equivalent function is `model.sel()`.

```
aictab(models)     # AICcmodavg function
model.sel(models) # MuMIn function
```

We've used AIC model selection before, but now that we are more familiar with what is going on under the hood take a moment to think through the different values returned by `aictab()`. Remember, the first two columns are `K`, the number of parameters in each model, and the AICc value, respectively. The third column, `Delta_AICc` (or ΔAICc) returns the difference in AICc between the model and the AIC-best model. Because AIC values are relative, only the relative differences are of interest. Ecological statisticians such as Burnham and Anderson provide some more specific guidelines for interpreting ΔAICc: models with ΔAICc<2 have all substantial support for being considered reasonable models, those with ΔAICc=2-7 have less support for being reasonable models, and models with ΔAICc=8-14 have relatively little support for all being reasonal models. If these guidelines still seem rather vague, that is intentional. Many people use hard ΔAICc cutoffs, much in the same manner as people use α to determine statistical significance. Burnham and Anderson have always recommended that ΔAICc be interpreted as a continuous measure of empirical support. **These guidelines naturally beg the question: what do we do if multiple models have "strong" empirical support for all being reasonable models? That is the focus of *multimodel inference* (more on that in a minute).**

Rounding out the `aictab` table. `AICcWt` (often denoted as ω_i) can be defined as model probabilities, specifically the probability that model i is the K-L-*best* model in the candidate model set. Another way to define them are as the probabilities of model i given the data. So far we haven't used these probabilities much - but now we'll explore them more.

w_i values are straightforward to derive by hand:

$$w_i = \frac{\exp(\text{-}0.5{*}\Delta_i)}{\sum_{i=1}^{I} \exp(\text{-}0.5{*}\Delta_i)}$$

where the weight (ω_i) of a any model is expressed as the exponent of the ΔAICc value the model i multiplied by -0.5, divided by the sum of the same term for *all* models. Model weights can also be calculated using the `Weights()` function from package `MuMIn`. `Weights()` requires a vector of AICc (or AIC) values. Since our models are stored in a list, we can use the `sapply()` function to *apply* a function to each list object, and then pass the resulting vector of AICc's to the `Weights()` function.

```
model.aiccs <- sapply(models,AICc)
Weights(model.aiccs)
```

Now calculate the weights of the 8th model by hand.

If we wanted to know what models make up 95% of the model support, we simply sum w_i until the cumulative sum is ≥ 0.95. Finally, the last column in the table is `LL`, the log-likelihood.

Notice also that using `model.sel()` from the `MuMIn` package returns much of the same information but also returns the parameter estimates for each model - this is a nice feature!

Quiz yourself: in the calculation of AICc, can you see the relationship between `K` and the `LL`? Do you and your table-mates now better understand what each column in the `aictab()` table represents?

Okay, now that we have made sure we understand things, interpret ΔAICc and ω_i to see that models with urban, and to a lesser extent, upland cover have the strongest support. The null model has virtually no support so we can conclude that land cover predictors greatly improve our ability to describe the variation in EIS home range size over and above the sex and tracking days variation. But wait! Multiple models have some degree of empirical support, in particular, two models have essentially equal probabilities of being the 'best' model. What do we do if we want to make inferences regarding the variables in our models? But more importantly, is there a way to incorporate the information about model support in a way that uses all of our models for making inferences. Yes! It's called *model averaging*.

3.3 Multimodel Inference and Model Averaging

Model averaging involves using the ω_i's to calculate weighted-average model predictions, parameter estimates, or standard errors. In this way, values from higher ranked models are given more weight and vice versa. Model averaging obviates the need to choose which models should be used for inference; all models can be used because those whose $\omega_i \approx 0$ will have little to no influence on the averaged value.

Before we proceed, let's illustrate the concept of a weighted average using an example from the field of landscape ecology. Say the area of several forest patches is measured (using aerial imagery and GIS) - some are very small patches and some very large patches. A simple average of patch size could be computed. Alternatively, the *area-weighted* patch size can be calculated which gives more weight to larger patches. Both measures have different meanings which are more appropriate in different applications. A weighting variable, `weight`, can be created by dividing each value of patch size, `size$size`, by the sum of all patch sizes. Notice that `weight` sums to one! To compute a weighted average, simply multiply each value of `size` by its weight and then sum these products. The unweighted average is about 43, the *area-weighted* average is about 79.

```
size <- data.frame(size=c(1,3,5,10,25,70,80,90,100))
mean(size$size)
size$weight <- size$size/sum(size$size)
sum(size$size*size$weight)
```

This same principle applies in model averaging. Fortunately, both the `AICcmodavg` and `MuMIn` packages have several useful functions for model averaging so we don't have to do the calculations by hand.

3.3.1 Model Averaging Predicted Values

We will start with model averaging predicted values because they are the simplest, both conceptually and computationally. For a given predicted value $\hat{y}_i$ from a given model, say m, we multiply $\hat{y}_i$ by w_m and then sum these products across all M models. Let's work a simple example by hand first. Say we are interested in the model averaged expected log home range size for males holding all other model covariates constant at their mean value, which is zero because covariates are z-score standardized (remember doing this at the beginning? Here is one of the times it is helpful!). We would do something like this:

```
l.hr <- rep(NA,8)                    # empty vector we will fill in
df <- data.frame(sex = "MALE",       # prediction for males
                 Sndays = 0,         # the n days
                 Surban1.50 = 0,     # the mean urban
                 Supland1.50 = 0,    # the mean upland
                 Swetland1.50 = 0) # the mean wetland
l.hr[1] <- predict(models[["Null"]],      newdata=df)
l.hr[2] <- predict(models[["Urb"]],       newdata=df)
```

```
l.hr[3] <- predict(models[["Up"]],        newdata=df)
l.hr[4] <- predict(models[["Wet"]],       newdata=df)
l.hr[5] <- predict(models[["Urb+Up"]],    newdata=df)
l.hr[6] <- predict(models[["Urb+Wet"]],   newdata=df)
l.hr[7] <- predict(models[["Up+Wet"]],    newdata=df)
l.hr[8] <- predict(models[["Urb+Up+Wet"]],newdata=df)
#exponentiate to get back on to the 'real' scale
HR <- exp(l.hr)
pred.df <- data.frame(l.hr=l.hr, HR = HR)
```

Notice an approximately 10% difference in expected male home range size across the model set. Let's add the ω_i's to the data frame, multiply each expected home range size by its corresponding ω_i, and then sum the products to obtain the *model-averaged expected male EIS home range size*. See how it differs from the arithmetic mean where all values have equal weights?

Quiz yourself: if your top model, m_{top}, had an AIC weight of one, $w_{top} = 1$, what would your model averaged parameter estimates be?

```
# Extract corresponding model weights for each prediction.
pred.df$w <- aictab(models)$AICcWt
# Calculate weighted mean home range size
sum(pred.df$HR * pred.df$w)

# Create a vector of equal weights
pred.df$equ <- 1/8             # Column of equal weights
sum(pred.df$HR * pred.df$equ) # Arithmetic mean home range size
mean(pred.df$HR)               # Arithmetic mean home range size
```

Actually, like most things in `R`, model-average expected male EIS home range sizes can be generated more quickly using either the `modavgPred()` function from the `AICcmodavg` package or the `predict()` function from the `MuMIn` package. We will use `modavgPred()` for this example for simplicity. The two major arguments for this function are a list of fitted models (`cand.set`) and a data frame containing the values used to make predictions (`newdata=`). The `newdata` must have the same column names as the data used to fit all the models - the `df` object from above will do.

```
modavgPred(models,newdata=df)
```

The function `modavgPred()` returns the model-averaged predicted value, the unconditional standard error (more on what unconditional means later), and a 95% Confidence Interval around the the predicted value.

What if we want to know how expected home range size changes as a function of one of our land cover types, say the amount of upland habitat around the home range changes? We simply replace the mean value of upland (zero in this case) with a vector of possible values for upland habitat. Let's look at expected home range size over a range of upland1.50 values from 0-0.90 (the approximate range of our observed data) and hold all other variables constant at their means. Let's also estimate separate predicted values for males and females. We won't worry about the other arguments in `modavgPred()` right now but they include a vector of model names (`modnames=`), a logical indicator of whether AICc should be used (`second.ord=`), and the confidence level for confidence intervals (`conf.level`).

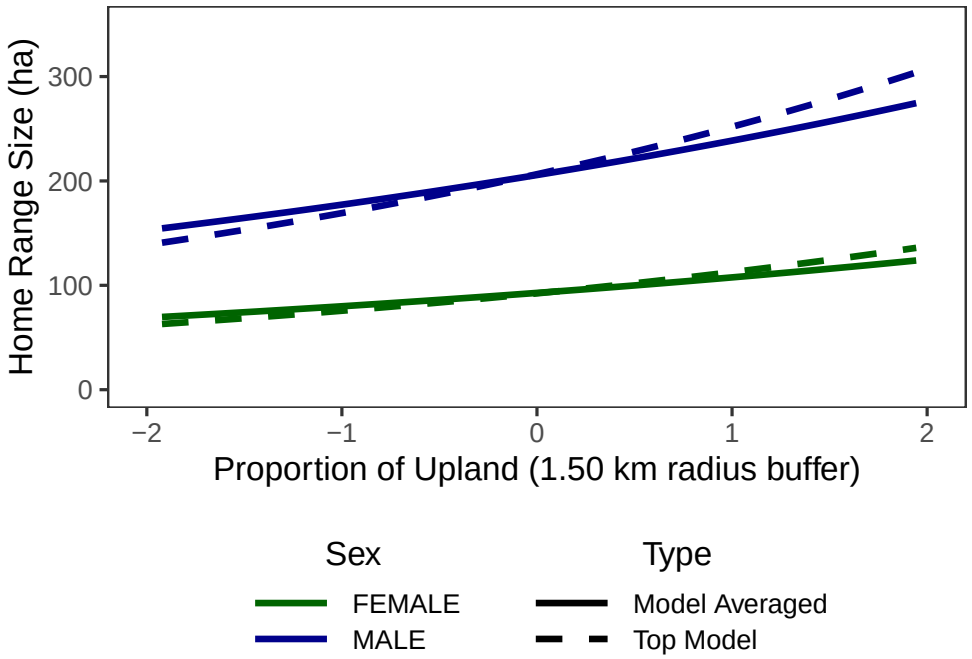
```
df2 <- data.frame(Supland1.50 = rep(seq(min(indigos$Supland1.50),
                                        max(indigos$Supland1.50),
                                        length=50),2),
                  sex = rep(c("MALE","FEMALE"),each=50),
                  Sndays = mean(indigos$Sndays),
                  Surban1.50 = mean(indigos$Surban1.50),
                  Swetland1.50 = mean(indigos$Swetland1.50))

MA_pred <- modavgPred(models,newdata=df2)
```

Remember that these values are on the log-scale and need to be exponentiated to see expected home range sizes on the scale of the raw data. Here is a graphical comparison of the model-averaged predicted home range size and predicted home range size from the top model. Try to create a similar figure yourself (don't worry about the bells and whistles). Here is some code to get you going:

```
#create intervals on the real scale
df2$pred <- exp(MA_pred$mod.avg.pred)
df2$LCI  <- exp(MA_pred$lower.CL)
df2$UCI  <- exp(MA_pred$upper.CL)
df2$Type <- "Model Averaged"
top_data <- df2[,1:5]
```

```
pred.values <- predict(models[["Urb+Up+Wet"]],
                       newdata=top_data,
                       se.fit=T)
pred.values <- interval(models[["Urb+Up+Wet"]],pred.values)
top_data$pred <- exp(pred.values[,"fit"])
top_data$LCI <- exp(pred.values[,"ci.low"])
top_data$UCI <- exp(pred.values[,"ci.hi"])
top_data$Type <- "Top Model"
new_data <- rbind(df2, top_data)
```



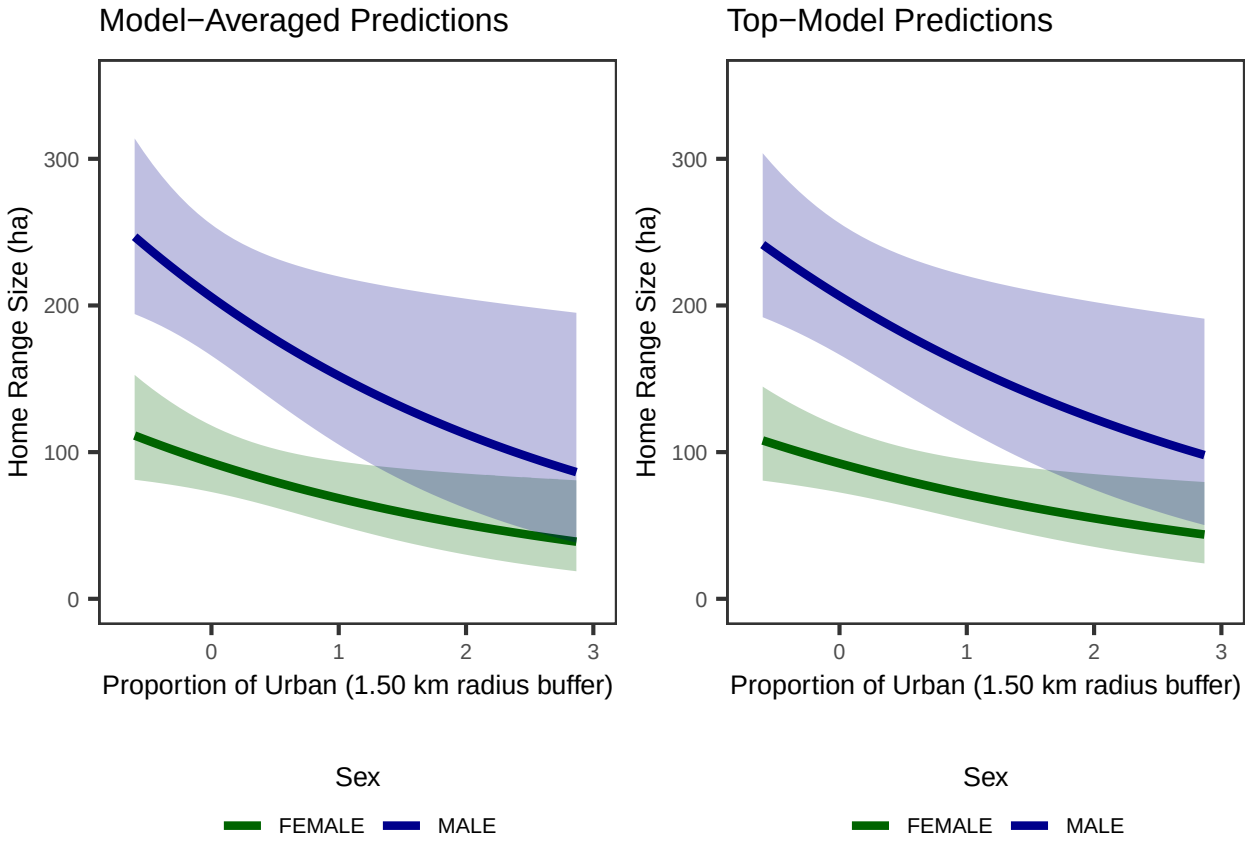
Unsurprisingly, the most important use of model-averaged values is for predictive purposes. Using all models for prediction, weighted by their degree of empirical support, generally leads to better predictive performance. For example, if you want to predict probability of species presence using occupancy models, model-averaged predictions may be more accurate than those made from a single model.

Quiz yourself: why do you think model averaged predictions often do a better job of predicting "reality"?

Model-averaged predictions can also be used to convey information about parameter effect sizes (e.g., what is the change in y over a specified range of X?). We could calculate model-averaged predictions for `urban1.50` and `wetland1.50` for each `sex` while holding all other continuous covariates constant at their means. Effect sizes can easily be communicated

graphically, as in the example above.

Of course, it is always desirable to add some estimate of the precision of our predictions, such as standard errors or confidence intervals. The previous graph can be modified to show model-averaged predicted values across a range `urban1.50`, but this time including the 95% Confidence Intervals of the predicted values. Again, see if you can reproduce something similar (minus bells and whistles), including the comparison of the predicted values and 95% CIs for the top model.



We might conclude based on this graph that there is a strong effect of sex on expected home range size, given that the CI do not entirely overlap. We might also conclude that urban has a strong negative effect on EIS home range size given that the expected home range size decreases by 65% across the range of urban values we observed. But we might also want to make inferences directly from the parameter estimates themselves, which leads to the next topic: *model averaging parameter estimates*.

3.3.2 Model Averaging Parameter Estimates

Model averaging parameter estimates or beta estimates follows the same general idea as model averaging predicted values (we will use the more general notation of $\hat{\theta}_i$ to refer to

parameter estimates): multiply the $\hat{\theta}_i$'s by their corresponding ω_i and then sum the products. However, there are two approaches for calculating model-averaged parameter estimates.

The first, and simplest, involves averaging the $\hat{\theta}_i$'s for *only* the models in which $\hat{\theta}_i$ appears. As an example, consider model-averaging $\hat{\beta}_{up1.50}$. From the `aictab` table and based on the model names, it's sen that `upland1.50` occurs in models 3, 5, 7 and 8. We need to extract the $\hat{\beta}_i$ from each model and then recalculate the w_i for each model to ensure they sum to one. Let's use the `Weights()` function from `MuMIn` to do this. Notice that when `MuMIn` is loaded, three functions are now masked from `AICcmodavg`. That is because each package contains functions with those names and `R` will use the function from the most recently loaded package. Functions from `AICcmodag` can still be accessed by adding the package name and a double colon in front of the function (e.g., `AICcmodavg::AICc`).

```
names(models)
aictab(models)
wi <- Weights(AICc(models[[3]],models[[5]],models[[7]],models[[8]]))
betai <- c(coef(models[[3]])['Supland1.50'],coef(models[[5]])['Supland1.50'],
           coef(models[[7]])['Supland1.50'],coef(models[[8]])['Supland1.50'])
betai
(MA_beta <- sum(wi * betai))     # Model-averaged estimate
coef(models[[7]])['Supland1.50'] # Top-ranked model estimate
coef(models[[5]])['Supland1.50'] # Second-ranked model estimate
```

Notice that the larger parameter values from models 3 and 7 have very little impact on the model-averaged parameter estimate because of their low ω_i, although they do act to pull the estimate up very slightly.

Model-averaged standard errors are calculated in a similar way, but with another twist. There are two sources of uncertainty when doing model-averaging: uncertainty in the estimates themselves (the standard error of $\hat{\beta}_i$), but also model uncertainty. You can think of the latter as the variation in $\hat{\beta}_i$ across all candidate models. Both sources of uncertainty must be accounted for or else our model-averaged standard errors will be biased low. To do this unconditional variances are computed. Unconditional simply means they are not conditional upon any given model in our candidate set. The formula for the unconditional variance is shown below:

$$\text{var}(\bar{\theta}) = \sum_{i=1}^{I} w_i(\text{var}(\hat{\theta}_i|g_i) + (\hat{\theta}_i - \bar{\theta}))$$

The term $\text{var}(\hat{\theta}_i|g_i)$ simply represents the variance of $\hat{\theta}_i$ in model g_i while the term $(\hat{\theta}_i - \bar{\theta})$ represents the variation in $\hat{\theta}_i$ across models. To go from unconditional variance to unconditional standard error you take its square root.

Let's calculate the unconditional standard error for `upland1.50` across all models containing `upland1.50` by hand, just to make sure its derivation is clear. As with calculating model-averaged parameter estimates, ω_i is recalculated based on only the models containing `upland1.50`. Variances for $\hat{\beta}_{up1.50}$ can be extracted from the variance-covariance matrix using `vcov()`. We will take the square root of the model-averaged variance to get the model-averaged standard error and then multiply this value by 1.96 to compute 95% CI.

```
names(models)
aictab(models)
wi <- Weights(AICc(models[[3]],models[[5]],models[[7]],models[[8]]))
betai <- c(coef(models[[3]])['Supland1.50'],coef(models[[5]])['Supland1.50'],
           coef(models[[7]])['Supland1.50'],coef(models[[8]])['Supland1.50'])
vari <- c(vcov(models[[3]])['Supland1.50','Supland1.50'],
          vcov(models[[5]])['Supland1.50','Supland1.50'],
          vcov(models[[7]])['Supland1.50','Supland1.50'],
          vcov(models[[8]])['Supland1.50','Supland1.50'])

#model averaged SE's
model_var <- (betai - sum(betai * wi))^2        # Var from model uncertainty
MA_SE <- sqrt(sum(wi * (vari  +  model_var)))   # Model-averaged SE
MA_beta - (1.96*MA_SE)
MA_beta  +  (1.96*MA_SE)

#top model SE's
SE <- sqrt(vcov(models[[8]])['Supland1.50','Supland1.50']) # global SE
coef(models[[8]])['Supland1.50'] - (1.96*SE)
coef(models[[8]])['Supland1.50'] +  (1.96*SE)
```

Notice how the model-averaged standard error is slightly larger than the standard error? This reflects the model selection uncertainty. Also note that the 95% CI overlap zero, implying that the effect of upland habitat does not have a "significant" effect on EIS home range size. However, Burnham and Anderson's analysis paradigm refrains from assigning "significance" to parameter estimates although the fact that the 95% CI includes zero does say something about the precision of the estimated effect size. Some researchers use CI to determine which

covariates should be used for making inference.

Fortunately, `AICcmodavg` (and `MuMIn`) provide functions that do all this work! Let's first use the `modavg()` function from `AICcmodavg`. `modavg()` requires a list of fitted model objects and the name of the parameter of interest. As with `modavgPred()`, there are other arguments available to modify different features of the calculations.

```
MAupland <- modavg(models,"Supland1.50")
MAupland
```

`modavg()` returns the revised AICc table with only those models containing our parameter of interest. It also provides the model-averaged parameter estimate, unconditional standard error, and 95% CI.

Now you might be asking, what about the models that do not contain our parameter of interest? Is it really appropriate to not incorporate them into our calculations of model-averaged parameter estimates and standard errors? This brings us to the second approach for calculating model-averaged parameter estimates. But before examining this approach, we need to discuss the issue of *model selection bias*, because this second approach is designed to account for such bias. Model selection bias is similar to *model selection uncertainty* but specifically refers to the consequences of this uncertainty on our model-averaged parameter estimates.

The solution, fortunately, is simple: the original formula for model-averaged parameter estimates is used, but now the average is taken across *all* models, and $\hat{\beta}_i$ is set to 0 if it does not appear in the model. This has the effect of "shrinking" model-averaged estimates towards zero and the degree of "shrinkage" depends on the degree of support for models containing that covariate. Model-averaged estimates calculated in this manner are typically referred to as the *full* estimates in contrast to *subset* estimates described earlier (this terminology follows that used in the `MuMIn` package). *Full* estimates are sometimes called *shrinkage* estimates because they behave similar to formal methods for estimating shrinkage parameter estimates.

Let's illustrate the differences between *full* and *subset* model-averaged parameter estimates using the `modavgShrink()` function from `AICcmodavg`. `modavgShrink()` has the same arguments as `modavg` and returns the same output. Because we expect the difference between the *full* and *subset* parameter estimates to differ the most for poorly-supported covariates, let's compute the model-averaged parameter estimates for wetland1.50.

```
modavg(models,"Swetland1.50")
modavgShrink(models,"Swetland1.50")
```

Notice that in the AICc table returned by `modavgShrink()`, the parameter estimate and standard error for `wetland1.50` has been replaced with 0 in models in which it is absent. As expected, the *full* estimate is smaller (44% smaller) than the *subset* estimate. The standard error and 95% CI for the *full* estimate is also larger than for the *subset* estimator. But what about the error message that was returned with `modavgShrink()`? Burnham and Anderson recommend that each covariate appear in an equal number of models when averaging across all models. The simplest way to equal inclusion of covariates across all models is with an *all-subsets* approach, although equality can be achieved using fewer candidate models as well. However, Burnham and Anderson do not elaborate on the reasoning behind this recommendation and some authors seem to suggest that *full* model-averaged parameter estimates should be used regardless of the equality of covariates among models. Furthermore, *full* model-averaged parameter estimates are conceptually equivalent to model-averaged predictions. Say that Model A contains the covariates X_1, X_2, and X_3 and Model B contains the covariates X_1, X_2, and X_4. A prediction from Model A implicitly assumes that $\hat{\beta}_4 = 0$, therefore $\hat{\beta}_4 \times X_4 = 0$, which is the way X_4 is dealt with when calculating *full* model-averaged parameter estimates. It is generally easy to develop a set of candidate models that ensures equality of covariates among models (particularly in exploratory or observational studies where an all-subsets approach is logical), yet we suggest still using *full* model-averaged parameter estimates where such estimates are of interest, regardless of the candidate models.

Quiz yourself: can you think of a scenario where *full* and *subset* estimates would be the same?

Before moving on, let's calculate model-averaged parameter estimates using `model.avg()` from `MuMIn`. `model.avg()` can take a list of fitted model objects and has a few different arguments from the `AICcmodavg` functions (see `?model.avg` for more details). `model.avg()` returns a matrix with the *full* and *subset* model-averaged parameter estimates. Notice how the estimates for `sexMALE` and `Sndays` are identical because they appear in *every* model. To obtain the standard errors (from which CI can be calculated), use `summary` on the `model.avg()` object as follows:

```
model.avg(models)
summary(model.avg(models))
```

3.3.3 Parameter Weights and Variable Importance

While model-averaged predictions and/or parameter estimates may be sufficient to evaluate the influence of our covariates on our response variable, there are two other approaches for

evaluating variable importance that we will discuss. The first is to calculate AIC parameter weights (ω). This can be calculated by summing the AIC weights (ω_i) for each parameter across all models in which that parameter appears. AIC parameter weights are only valid when all covariates appear in an equal number of models. Again, while an all-subsets approach ensures that each covariate appears in an equal number of models, there are other ways to ensure this equality. AIC parameter weights provide an indication of the *relative* importance of the covariates. Another way to think of ω is as the probability that a particular covariate is included in the "best" model. Looking at the original AICc table, `urban1.50` appears in the top four models and the sum of ω_i's across these models is 0.95, which is also obtainable from the `Cum.Wt` column. The covariate `upland1.50` appears in the first, second, fifth, and sixth models and the sum of ω_i across these models is 0.74. `AICcmodavg` and `MuMIn` each have functions to return AIC parameter weights. Unfortunately, both functions are named `importance()`. If both packages are loaded, `R` will recognize the function from the last package loaded. The `importance()` function from `MuMIn` is slightly simpler to use but examples of both function's are provided below. From the results, we see that `urban1.50` has the highest relative importance, followed by `upland1.50`, and then `wetland1.50`. We cannot evaluate the relative importance of sex and number of days because we included them in every model. It is important to remember that AIC parameter weights are merely *relative* rankings of our covariates. They do not convey information about the absolute importance of the covariates or the amount of variation explained (e.g., R^2).

```
aictab(models)
AICcmodavg::importance(models, parm="Surban1.50") # AICcmodavg
MuMIn::importance(models) # MuMIn
```

AIC parameter weights have also been the subject of some controversy. This stems partly from the fact that ω_i are specific to a particular model, not the covariates within that model, and partly from the fact that AIC parameter weights do not necessarily correspond to amount of variation explained by a particular covariate. For example, X_1 may have a high ω because, when combined with other covariates, X_1 appears in one or more highly ranked models. Yet a model with X_1 may have very poor performance. However, when interpreted properly (i.e., as the probability of a covariate occurring in the top-ranked model), AIC parameter weights can still provide a valid and useful measure of relative variable importance.

Ok, time to *REST*! that was a lot to take in over the course of one lab. Do you think you could (generally) explain the rationale, process, and interpretation of model averaging to your neighbor? Thankfully, you can always come back to me (this is the **PDF** talking here) as a refresher! For this week's lab, turn in a *one* paragraph synopsis of when model

averaging might be useful.