

Week 4: Model selection

bbmle For these exercises, and most of the exercises for the remainder of the term, you'll want to use some optional but convenient functions—mainly the functions `AICtab()` and `AICctab`, but also soon `mle2()`—in a package called `bbmle`. You likely do not have this package on your computer yet. To install it, find the “Package Installer” command in one of R's menus—I think it's in a different place on Macs and Windows—and then you can find it in the list of available packages for download and install it, without leaving the application. Once it is installed in R, you can make its functions available with the command `library(bbmle)`. In Exercise 1, below, I'll walk you through the first uses of this library.

coefstab I am also posting for this week a function I wrote a few years ago that makes tables of estimates, for comparing models. You'll use this function in Exercise 2. You can make the `coefstab()` function available by loading the contents of `coefstab.r` into R. You won't see any output; it just reads the code into memory, so it'll be available when you want to use it.

Exercise 1: “Testing” for a single effect

Again load the class height and weight data into R. We're going to analyze it using AIC, as a means of illustrating how to use model comparison on even very simple data sets.

(a) Fit two linear models to the height and weight data. First, fit the model that attempts to predict height using weight. Second, fit the model that assumes the mean height is a constant. Use `lm()` for both of these, saving the fit models into objects, using the name `m1` for the height \sim weight model, and `m0` for the intercept-only model.

(b) Now plot the mean predictions of both models. First make a scatter plot of the raw height \sim weight values, with weight on the horizontal axis. Second, use `abline()` to superimpose both fit models over these data. The function `abline()` does understand the `col` option, so you can try to make each prediction line a different color.

(c) Now simulate observations from each model and plot them. I did this sort of thing in the Week 3 lecture notes. I'll walk you through this one, and you'll do it yourself in later exercises. Generate 100 random weight values between the observed minimum and maximum with:

```
fakew <- sample( min(d$weight):max(d$weight) , 100 , replace=TRUE )
```

These are the simulated weights that we will predict heights for.

Let's do the intercept-only model first. Extract the value of the intercept (α) with:

```
k0 <- coef(m0)
```

Compute the approximate standard deviation of the distribution, σ , with:

```
sd0 <- sd( residuals( m0 ) )
```

You can simulate heights now with:

```
hm0 <- rnorm( 100 , mean=k0["(Intercept)"] , sd=sd0 )
```

The second model works the same way.

```
k1 <- coef(m1)
sd1 <- sd(residuals(m1))
hm1 <- rnorm( 100 , mean=k1["(Intercept)"] + k1["d$weight"] * fakew , sd=sd1 )
```

Note that in both cases, you can refer to each specific parameter estimate by its name. You can remind yourself of the names by just typing `k0` or `k1` on the command line, or by using the `summary()` function.

I leave it as an exercise for the student to plot both sets of predictions on a graph with weight on the horizontal axis and height on the vertical axis. (You can use the function `points()` to plot points, like `lines()` plots lines.) Color each set of points differently, so you can tell them apart. What additional information does simulating the model predictions give you, that you didn't get from plotting the mean predictions in part (b)?

(d) Okay, now we're ready to compare these two models, using AIC. Type:

```
library(bbmle) # this line just loads the AICtab function so R can use it
AICtab( m0 , m1 , base=TRUE , weights=TRUE )
```

How do you interpret the output, given your knowledge of AIC and Akaike weights? To compute the bias-adjusted AIC_c , use:

```
AICctab( m0 , m1 , base=TRUE , weights=TRUE , nobs=28 )
```

What has changed, and why? What does this tell you about the direction of bias that AIC_c corrects for? That is, does the bias favor models with more parameters or fewer?

Exercise 2: Urban Foxes 2: Electric Boogaloo

Let's revisit the urban foxes data from last week and use model comparison on it.

(a) Fit the following models:

1. `WEIGHT ~ 1`
2. `WEIGHT ~ AVFOOD`
3. `WEIGHT ~ AVFOOD + GSIZE`
4. `WEIGHT ~ AVFOOD + GSIZE + AREA`

(b) Compare the predictions of each model, with respect to the relationship between WEIGHT and AVFOOD. That is, plot the mean predicted WEIGHT across values of AVFOOD, for each model you fit above. For models with more variables than just WEIGHT and AVFOOD, hold the other prediction variables at their mean values in the real data. How would you describe the predicted relationship between WEIGHT and AVFOOD, using plain English, in each case?

(c) Compare the models from (a), using AIC_c . What do you conclude about each potential prediction variable, from the results?

(d) Finally, use the Akaike weights from the `AICctab` table to average the models, to produce a joint set of predictions that accounts for model uncertainty. To do this, instead of using the parameter estimates of each model directly, average the estimates across models, using the weight of each model to compute the weighted average. This procedure produces a single set of model-averaged estimates, and you can then plug these into the linear model and produce model-averaged predictions.

To make comparing parameter estimates a bit easier, use the `coefstab()` function that I supply on the website this week. This function makes a table of estimates. Each column is a parameter name and each row is a model. For example, using `coefstab()` on the Exercise 1 models:

```
> coefstab(m0,m1)
  (Intercept)  d$weight
m0      68.20536         NA
m1      51.42838  0.1051377
```

Where there is an “NA,” you want to pretend the parameter estimate is actually zero (0), because by not including the weight predictor in `m0`, you effectively assumed that the beta value was exactly zero. (This is known as a set of “nested” models.) So to average the estimates above, weight each estimate by the Akaike weight of each model, add them together to get average estimates, and then use those in a new model to produce predictions. It works the same way for the foxes data, but with more models and parameters.