

# Week 4: Model selection SOLUTIONS

## Exercise 1: “Testing” for a single effect

(a) To fit the models:

```
m0 <- lm( d$height ~ 1 )  
m1 <- lm( d$height ~ d$weight )
```

The first corresponds to the model:

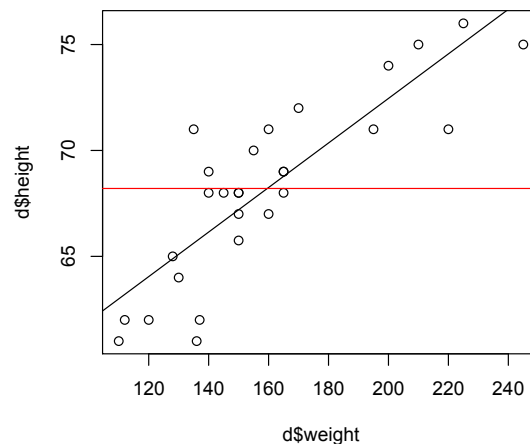
$$h_i \sim \mathcal{N}(\alpha, \sigma),$$

and the second corresponds to:

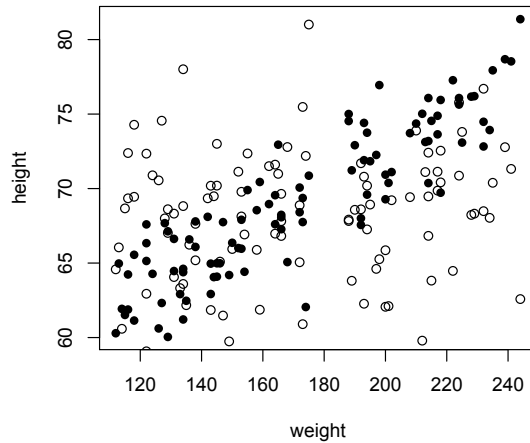
$$h_i \sim \mathcal{N}(\alpha + \beta, \sigma).$$

(b)

```
plot( d$height ~ d$weight )  
abline(m1)  
abline(m0,col="red")
```



(c) The extra info we get from the plot below, is to that the variance is much bigger for the intercept-only model (the white points).



(d)

```
> library(bbmle)
> AICtab( m0 , m1 , base=TRUE , weights=TRUE )
  AIC  df dAIC  weight
m1 130.4 3   0.0 1
m0 163.8 2  33.4 <0.001
> AICctab( m0 , m1 , base=TRUE , weights=TRUE , nobs=28 )
  AICc  df dAICc  weight
m1 131.4 3   0.0 1
m0 164.3 2  32.9 <0.001
```

The model including weight is far better, in terms of out of sample prediction. It dominates the Akaike weight, using either AIC measure. The AICc measure makes the distance between the models smaller (dAIC goes from 33.4 down to 32.9), suggesting that the bias that AICc corrects for favors complex models. As the number of parameters approaches the number of data points, un-corrected AIC will tend to strongly favor more complex models.

## Exercise 2: Urban Foxes 2: Electric Boogaloo

(a) Simple enough to fit the models at this point:

```
m0 <- lm( d$WEIGHT ~ 1 )
m1 <- lm( d$WEIGHT ~ d$AVFOOD )
m2 <- lm( d$WEIGHT ~ d$AVFOOD + d$GSIZE )
m3 <- lm( d$WEIGHT ~ d$AVFOOD + d$GSIZE + d$AREA )
```

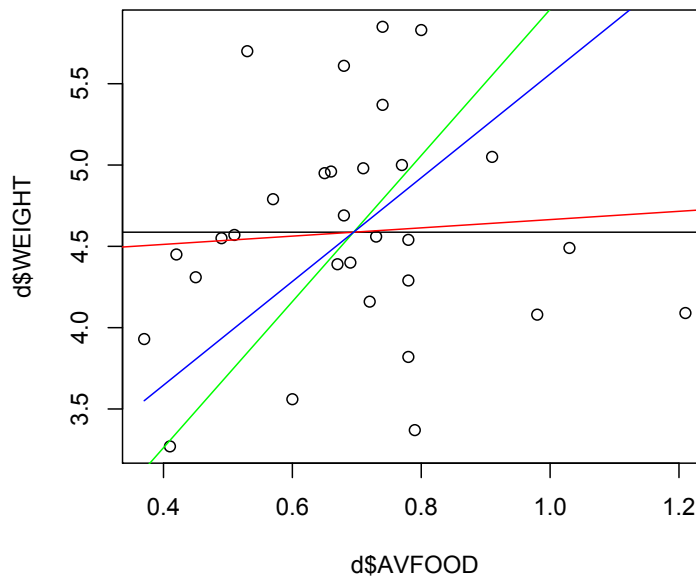
(b) One way to go about this is to make one scatter of WEIGHT and AVFOOD and then plot all the regression lines on top of it. For the first two models, our old friend `abline()` will

work fine. But for the last two models, it won't. Instead, you need to extract the parameter estimates and use the mean values of those variable you aren't plotting. Otherwise, the line won't be at the right height on the plot. In a multivariate model, un-plotted variables are affecting predictions, because they are in the regression equation always.

Here's how I did it:

```
plot( d$WEIGHT ~ d$AVFOOD )
abline(m0)
abline(m1,col="red")
# now can't use abline, because of extra coefficients
k <- coef(m2)
curve( k["(Intercept)"] + k["d$AVFOOD"]*x + k["d$G$SIZE"]*mean(d$G$SIZE) ,
      from=min(d$AVFOOD) , to=max(d$AVFOOD) ,col="green" , add=TRUE )
k <- coef(m3)
curve( k["(Intercept)"] + k["d$AVFOOD"]*x + k["d$G$SIZE"]*mean(d$G$SIZE) +
      k["d$AREA"]*mean(d$AREA) ,
      from=min(d$AVFOOD) , to=max(d$AVFOOD) ,col="blue" , add=TRUE )
```

This gives us:



As you can tell from reading the code, the black line is m0, the red is m1, the green is m2, and the blue is m3.

Now for plain English interpretations. In the intercept model, there is no true relationship between WEIGHT and AVFOOD; knowing AVFOOD tells you nothing about WEIGHT. In m1, knowing AVFOOD tells you only a little bit about WEIGHT; the relationship is positive, ever so slightly, so it seems more food has a mild effect on fox body mass. In m2,

now we see a very strong relationship between AVFOOD and WEIGHT: AVFOOD strongly predicts larger fox body masses. In the final model, the relationship between AVFOOD and WEIGHT is still strongly positive, but a bit milder than in the previous model.

(c) This will do the job:

```
library(bbmle)
AICctab( m0 , m1 , m2 , m3 , base=TRUE , weights=TRUE , nobs=30 )
```

```
      AICc df dAICc weight
m2 53.8 4    0.0  0.50936
m3 53.9 5    0.1  0.48800
m0 64.8 2   11.1  0.00201
m1 67.2 3   13.4  < 0.001
```

The model containing AVFOOD and GSIZE is the best, but only slightly better than the model containing all three, AVFOOD and GSIZE and AREA. The other two models have extremely low Akaike weights. This suggests any good model here will contain both GSIZE and AVFOOD. Adding AREA might help, but it's uncertain.

(d) After getting the `coefstab` code into R, you can see the table of estimates with:

```
> coefstab(m0,m1,m2,m3)
      (Intercept)  d$AVFOOD    d$GSIZE    d$AREA
m0      4.587000         NA         NA         NA
m1      4.410292  0.2542557         NA         NA
m2      3.988568  4.4920211 -0.6526352         NA
m3      4.002401  3.1888929 -0.6833843  0.3468955
```

The weights of each model come from the table you produced in part (c). So in principle all you need to do is multiple each estimate by the weight corresponding to its model, and then sum down the columns to get model averaged estimates. Remember to treat NA as zero (0).

You can also automate this a bit. Here's the code I used. First, store the table above in an object:

```
dtab <- coefstab(m0,m1,m2,m3)
```

Now replace all the NA's with zero (0):

```
# replace NAs with zeros
for ( i in 1:nrow(dtab) ) {
  for( j in 1:ncol(dtab) ) {
    dtab[i,j] <- ifelse( is.na(dtab[i,j]) , 0 , dtab[i,j] )
  }
}
```

Your `dtab` should now look like:

```
> dtab
      (Intercept) d$AVFOOD    d$GFSIZE    d$AREA
m0      4.587000 0.0000000  0.0000000 0.0000000
m1      4.410292 0.2542557  0.0000000 0.0000000
m2      3.988568 4.4920211 -0.6526352 0.0000000
m3      4.002401 3.1888929 -0.6833843 0.3468955
```

Now I'll add a new column to this table that contains the Akaike weights:

```
# add weights to dtab
dtab$weight <- AICctab( m0 , m1 , m2 , m3 , base=TRUE , weights=TRUE ,
  nobs=30 , sort=FALSE )$weight
```

That line is actually making an AICctab, but only returning the column named `weight`. Note that I turned off sorting with `sort=FALSE`, so that these weights are in the same order as the models in the `coef` table. Now we have:

```
> dtab
      (Intercept) d$AVFOOD    d$GFSIZE    d$AREA    weight
m0      4.587000 0.0000000  0.0000000 0.0000000 0.0020129096
m1      4.410292 0.2542557  0.0000000 0.0000000 0.0006282504
m2      3.988568 4.4920211 -0.6526352 0.0000000 0.5093633933
m3      4.002401 3.1888929 -0.6833843 0.3468955 0.4879954467
```

Finally, multiple each parameter column by the weight column, and sum down each column of products:

```
# average down columns, using weights
weighted.coefs <- rep(0,4)
for ( the.col in 1:4 ) {
  weighted.coefs[the.col] <- sum( dtab[,the.col] * dtab[,5] )
}
```

And the weighted coefficients are:

```
> weighted.coefs
[1] 3.9967879 3.8443961 -0.6659169 0.1692834
```

These are, in order, the intercept, AVFOOD coefficient, GSIZE coefficient, and AREA coefficient. You can put names on the elements of this vector with:

```
names(weighted.coefs) <- colnames(dtab)[1:4]
```

So finally:

```
> weighted.coefs
(Intercept)    d$AVFOOD    d$GFSIZE    d$AREA
 3.9967879    3.8443961   -0.6659169    0.1692834
```

Notice that the model weighted estimate of AREA is about half of the estimate in m3, because the Akaike weight of m3 is about 0.5. Model uncertainty reduces the influence of AREA in the averaged model, which takes model uncertainty into account. It has very little effect on the estimates of AVFOOD and GSIZE, because those estimates don't vary much across models with the vast majority of the Akaike weight.