Armidale Practical Materials

Michael Morrissey

02 February, 2020

This document contains worked answers. Please work from the questions-only document in the first instance!

The data

Soay sheep are a neolithic breed of sheep that have been unmanaged in the St Kilda (Hebrides, Scotland) archipelago since prehistoric times. Since 1985, an individual-based study has collected life history, pedigree, morphological, phenological and parasitological data on the Soay sheep population in Village Bay on Hirta, St Kilda. The example dataset contain records of selected phenotypes and fitness measures for female Soay lambs with normal horns from selected cohorts that had relatively high first year survival. The phenotypic traits are all morphological measures for mass (`MASS`, kg), right hind metatarsal length (`HINDLEG`, mm) and horn length (`HORNLEN`, mm) measured in August of the first year of life (age: approximately 4 months). Fitness (component) measures are first year survival (`SURV`) coded as whether or not an individual survived its first winter (1: survived, 0: died), and total lifetime reproductive success (`LRS`), calculated as the number of live births recorded for each ewe throughout her life.

Here is code that might be useful for reading from the data file, assuming that it is in your working directory:

```
d<-read.table("./soay_ewe_lamb_selection_data.csv",sep=',',header=TRUE)
```

```
str(d)
```

```
## 'data.frame':    213 obs. of  6 variables:
##  $ ID     : int  2378 2110 2181 2248 2293 2311 2314 2323 2339 2351 ...
##  $ MASS   : num  15 10.2 11.4 14.3 12.2 15.7 14.4 16.6 16 13.5 ...
##  $ HINDLEG: int  168 144 161 167 165 171 165 176 169 156 ...
##  $ HORNLEN: int  101 65 81 19 92 106 100 120 72 85 ...
##  $ SURV   : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ LRS    : int  3 0 0 0 0 1 0 15 5 2 ...
```

Before setting off on selection analyses, you might want to explore the data a bit. Have a brief look at the means and variability of each trait. Histograms might be useful.

Univariate selection

(1) What is the mean mass in the unselected population, and among those individuals that survive their first year?

```
# mean before selection
mean(d$MASS)
```

```
## [1] 12.34742
```

```
# mean after selection (alternative, but equivalent calculations)
mean(d$MASS[which(d$SURV==1)])
```

```
## [1] 12.73918
```

```
weighted.mean(d$MASS,d$SURV)
```

```
## [1] 12.73918
```

```
with(subset(d,SURV==1),mean(MASS))
```

```
## [1] 12.73918
```

(2) What is the selection differential for mass acting through first year survival?

```
S_mass<-mean(d$MASS[which(d$SURV==1)]) - mean(d$MASS)
S_mass
```

```
## [1] 0.3917634
```

(3) Convert the selection differential that you just calculated into mean-standardised and variance-standardised differentials.

```
# mean-standardised
S_mass/mean(d$MASS)
```

```
## [1] 0.03172837
```

```
# variance-standardised
S_mass/sd(d$MASS)
```

```
## [1] 0.1790815
```

(4) In the previous question, you could have worked out what operation (e.g., multiplication, division), with the mean and standard deviation of mass, was necessary to apply to the selection differential, in order to make the re-scalings. Alternatively, you could have re-scaled the phenotype and then re-calculated the selection differentials on those re-scaled phenotypes. Whichever you didn't do above, do it now to verify your initial calculation.

```
# re-scale phenotype
d$MASS_mean<-d$MASS/mean(d$MASS)
d$MASS_sd<-d$MASS/sd(d$MASS)

# the mean- and unit variance-standardised differentials
weighted.mean(d$MASS_mean,d$SURV)-mean(d$MASS_mean)
```

```
## [1] 0.03172837
```

```
weighted.mean(d$MASS_sd,d$SURV)-mean(d$MASS_sd)
```

```
## [1] 0.1790815
```

(5) What is the selection gradient for mass, acting through first year survival (do the calculation without implementing a linear regression, i.e., without using the `lm()` function).

```
beta_mass<-S_mass/var(d$MASS)
beta_mass
```

```
## [1] 0.08186105
```

(6) Check that your calculation of the selection differential for mass (via first year survival) agrees with the OLS-based method (i.e., from Lande and Arnold 1983).

```
d$w_s<-d$SURV/mean(d$SURV)
summary(lm(w_s~MASS,data=d))$coefficients
```

```
##                Estimate Std. Error    t value     Pr(>|t|)
## (Intercept) -0.01554034 0.18270634 -0.0850564 9.322972e-01
```

```
## MASS         0.08224718 0.01457126  5.6444812 5.302175e-08
```

(7) What are the units of the selection gradient that you just calculated?

The units are kg^{-1} (for the model above). If you variance-standardised (which has somehow come to be considered a feature of the L&A analysis, if not part of the definition of selection gradients themselves, which is unfortunate), then the units would be σ_z^{-1}

(8) Calculate the selection differential and gradient for hind leg length (also acting through first year survival)?

```
S_leglen<-mean(d$HINDLEG[which(d$SURV==1)]) - mean(d$HINDLEG)
S_leglen
```

```
## [1] 1.518244
```

```
beta_leglen<-S_leglen/var(d$HINDLEG)
beta_leglen
```

```
## [1] 0.01752382
```

(9) Which is stronger, selection of hind leg length or mass?

If we want to think of the strength of selection in terms of the steepness of the fitness function, we will need to answer the question using selection gradients. There is no sense in comparing mm and kg, so we will want to use one of the standardisations:

```
# mean and variance standardised selection gradients for mass
beta_mass*mean(d$MASS)
```

```
## [1] 1.010773
```

```
beta_mass*sd(d$MASS)
```

```
## [1] 0.1790815
```

```
#mean and variance standardised selection gradients for leg length
beta_leglen*mean(d$HINDLEG)
```

```
## [1] 2.762347
```

```
beta_leglen*sd(d$HINDLEG)
```

```
## [1] 0.1631117
```

So, which is stronger? The answer is ambiguous. The variance standardised selection gradients say that over the range of phenotype expressed in the population, expected fitness changes by about the same amount. But a variance standardised gradient doesn't really parse out the effect of steepness of the fitness function from the variability of the trait. So, I think that mean standardisation is preferable here. A 1% increase in hind leg length corresponds to almost three times as much increase in fitness as does a 1% increase in mass. So, for the strength of selection, where by "strength" we mean the slope fo the fitness function, I think we can conclude that selection is stronger for leg length, at least insofarr as we might want a measure that is independent of the variability in the population.

However, we might want to think about the strength of selection in terms of how much the mean phenotype changes. This is a notion of the strength of selection to which both degree of trait variability, and the steepness of the trait-fitness function contribute. This is fine (well, it is wrong if you go making conclusions about the steepness of the fitness function based on such, which people do all the time). Here are the selection differentials, in both standardisations:

```
# mean and variance standardised selection differentials for mass
S_mass/mean(d$MASS)
```

```
## [1] 0.03172837
```

```
S_mass/sd(d$MASS)
```

```
## [1] 0.1790815
```

```
# mean and variance standardised selection differentials for leg length
S_leglen<-weighted.mean(d$HINDLEG,d$SURV)-mean(d$HINDLEG)
S_leglen/mean(d$HINDLEG)
```

```
## [1] 0.009631462
```

```
S_leglen/sd(d$HINDLEG)
```

```
## [1] 0.1631117
```

You should recognise these calculations (for mass, anyhow) from question 9. Note also that, for univariate selection, the values of the selection gradient and differential coincide, so those should agree with the ones in the first part of the answer.

Variance standardisation indicates that the shift in the mean, in relation to the range of variability for each trait before selection, is similar. However, selection changes the mean of mass by about 3% of its pre-selection value, whereas it only changes the mean of hind leg length by about 1%.

A final way of thinking about the strength, that draws both on the amount of variation and the steepness of the fitness function, is to think about the amount of variation in fitness resulting jointly from phenotypic variation and fitness function steepness. This quantity (as the SD of relative fitness that can be associated with the trait) comes from the variance standardised measures (for either gradients or differentials, in a univariate analysis). According to this notion of the strength of selection, selection is similar for mass and hind leg length.

Have a good think about the strengths of selection before reading this blurb. The reason why the comparison about the strength of selection is so interesting for leg length and mass is that the two traits have very different variability, in relation to their means. It all comes down to having a feel for the traits, in this case, some knowledge of their means and variances. Thinking about the strength of selection is hard, but it may have been harder than necessary if you didn't take the instruction to explore the data seriously. Here are some key things that could help:

```
# variance of each trait
var(d$MASS)
```

```
## [1] 4.785713
```

```
var(d$HINDLEG)
```

```
## [1] 86.63885
```

```
# standard deviations
sd(d$MASS)
```

```
## [1] 2.187627
```

```
sd(d$HINDLEG)
```

```
## [1] 9.307999
```

```
# means
mean(d$MASS)
```

```
## [1] 12.34742
```

```
mean(d$HINDLEG)
```

```
## [1] 157.6338
```

```
# coefficients of variation
sd(d$MASS)/mean(d$MASS)
```

```
## [1] 0.1771728
```

```
sd(d$HINDLEG)/mean(d$HINDLEG)
```

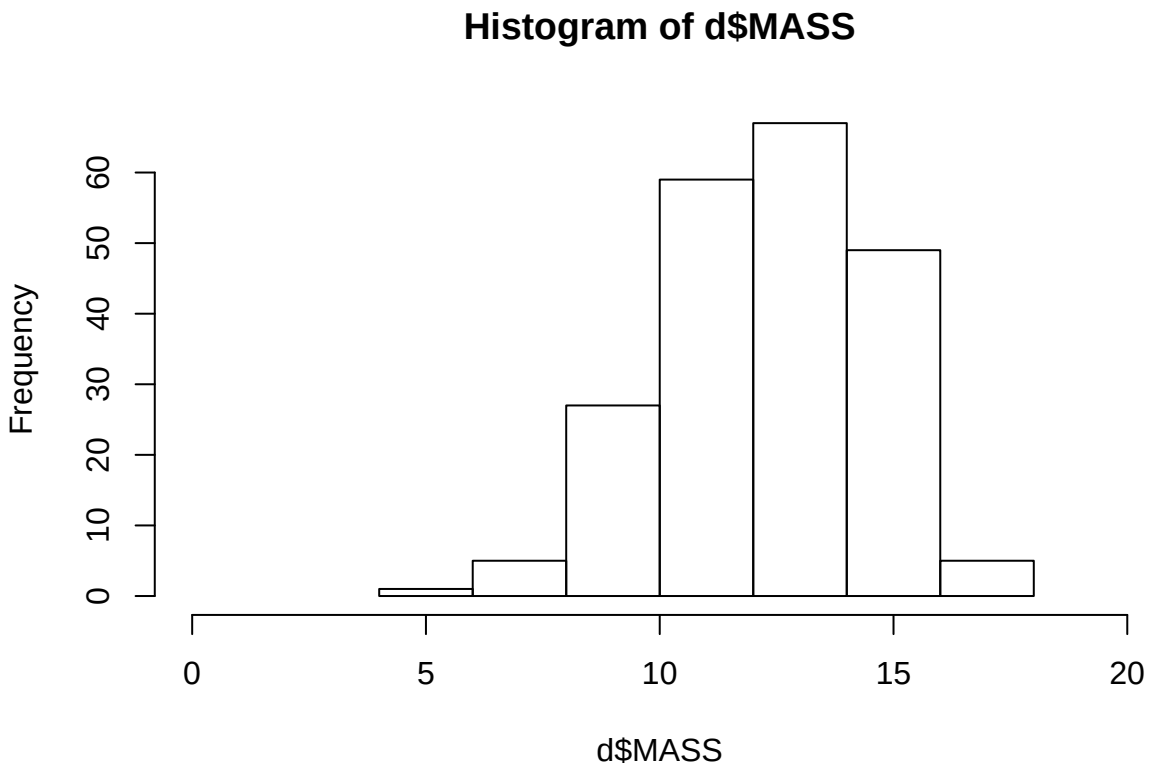
```
## [1] 0.05904824
```

Histograms of the two traits, roughly manipulated to be more helpful about both the mean and the variability:

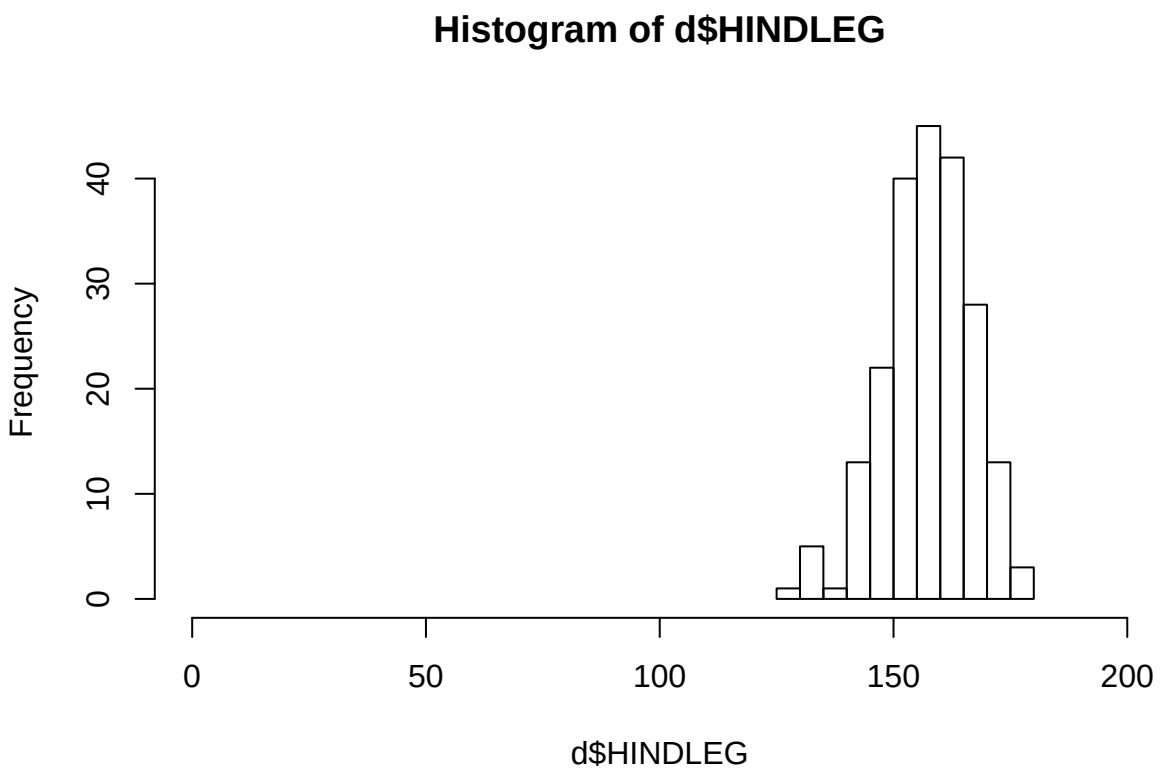
```
par(c(1,2))
```

```
## NULL
```

```
hist(d$MASS,xlim=c(0,20))
```




```
hist(d$HINDLEG,xlim=c(0,200))
```



(10) How much does first year viability selection change the variance of mass and leg length?

```
delta_v_mass<-var(d$MASS[which(d$SURV==1)])-var(d$MASS)
delta_v_mass
```

```
## [1] -0.4050804
```

```
delta_v_leglen<-var(d$HINDLEG[which(d$SURV==1)])-var(d$HINDLEG)
delta_v_leglen
```

```
## [1] -10.80328
```

(11) How much does first year viability selection change the variance of mass and leg length, over and above the effect of strictly directional selection to change the variance?

```
C_mass <- var(d$MASS[which(d$SURV==1)])-var(d$MASS) + S_mass^2
C_mass
```

```
## [1] -0.2516018
```

```
C_leglen <- var(d$HINDLEG[which(d$SURV==1)])-var(d$HINDLEG) + S_leglen^2
C_leglen
```

```
## [1] -8.498218
```

Multivariate selection

(12) Calculate the multivariate selection gradients of body mass, hind leg length, and horn length, acting through first winter viability.

I didn't specify directional or quadratic. So here it is with just directional, and with directional and quadratic gradients

```
m1<-lm(w_s~MASS+HINDLEG+HORNLEN,data=d)
summary(m1)$coefficients
```

```
##                 Estimate  Std. Error    t value    Pr(>|t|)
## (Intercept) -0.386031420 0.770316002 -0.5011338 0.616804323
## MASS         0.086983651 0.029191168  2.9797934 0.003225998
## HINDLEG      0.004259164 0.006665630  0.6389739 0.523539539
## HORNLEN     -0.004339948 0.001653272 -2.6250664 0.009303315
```

```
cent<-function(x){x-mean(x)}
d$MASS_c<-cent(d$MASS)
d$HINDLEG_c<-cent(d$HINDLEG)
d$HORNLEN_c<-cent(d$HORNLEN)
```

```
m2<-lm(w_s~MASS_c+HINDLEG_c+HORNLEN_c+I(0.5*MASS_c^2)+I(0.5*HINDLEG_c^2)+I(0.5*HORNLEN_c^2)+
          I(MASS_c*HINDLEG_c)+I(MASS_c*HORNLEN_c)+I(HINDLEG_c*HORNLEN_c),data=d)

# directional gradients
summary(m2)$coefficients[2:4,]
```

```
##               Estimate  Std. Error   t value    Pr(>|t|)
## MASS_c     0.075806858 0.030392651  2.494250 0.013420501
## HINDLEG_c  0.008239526 0.007002768  1.176610 0.240729089
## HORNLEN_c -0.005734195 0.002122237 -2.701958 0.007476474
```

```
# quadratic gradients
summary(m2)$coefficients[5:7,]
```

```
##                          Estimate   Std. Error   t value   Pr(>|t|)
## I(0.5 * MASS_c^2)    -0.073115871 0.0376441092 -1.942292 0.05348521
## I(0.5 * HINDLEG_c^2) -0.003813827 0.0018639663 -2.046082 0.04203760
## I(0.5 * HORNLEN_c^2) -0.000111997 0.0001034885 -1.082217 0.28044005
```

```
# correlational gradients
summary(m2)$coefficients[8:10,]
```

```
##                               Estimate   Std. Error    t value   Pr(>|t|)
## I(MASS_c * HINDLEG_c)     0.0145131652 0.0071695511  2.0242781 0.04425177
## I(MASS_c * HORNLEN_c)     0.0032937124 0.0019045594  1.7293829 0.08526073
## I(HINDLEG_c * HORNLEN_c) -0.0003415577 0.0004199518 -0.8133261 0.41698401
```

(13) Do our conclusions about the strength of directional selection on mass and leg length change when we consider the direct component of selection, rather than the overall association?

```
# a slight leg up on getting the vectors of the mean and sds that you'll need (hint, hint)
z<-d[,c("MASS","HINDLEG","HORNLEN")]
mu<-apply(d[,c("MASS","HINDLEG","HORNLEN")],2,mean)
sigma<-apply(d[,c("MASS","HINDLEG","HORNLEN")],2,sd)
```

```
beta<-coef(m1)[2:4]

# mean standardisation
beta*mu
```

```
##       MASS    HINDLEG    HORNLEN
##  1.0740235  0.6713882 -0.3593803
```

```
# variance standardisation
beta*sigma
```

```
##       MASS    HINDLEG    HORNLEN
##  0.1902878  0.0396443 -0.0939258
```

So, direct selection of mass and hind leg length is roughly in magnitude in the sense of the steepness of the fitness function, with a 1% increase in fitness with a 1% increase in mass, and abotu a 0.67% increase in fitness for a 1% increase in hind leg length. However, because of the much lower variability of hind leg length, relative to mass, direct selection of mass is responsible for much more variation in fitness than is hing leg length.

(14) [Advanced] Say you wanted to use a GLM to characterise the function relating mass, hind leg length, and horn length to fitness, and you particularly wanted to then recover the selection gradient estimate that belong to that specific model. Say your model was a binomial GLM, like this:

```
glm_lambs<-glm(SURV~MASS+HINDLEG+HORNLEN,data=d,family="binomial")
summary(glm_lambs)$coefficients
```

```
##                Estimate Std. Error    z value    Pr(>|z|)
## (Intercept) -5.89665169 4.38857578 -1.3436367 0.179065880
## MASS         0.58621352 0.18654017  3.1425592 0.001674778
## HINDLEG      0.02339123 0.03820471  0.6122604 0.540365476
## HORNLEN     -0.03863961 0.01428608 -2.7047035 0.006836539
```

And the expected fitness of all indivdiuals (according to the GLM), and the corresponding mean fitness of the popualtion would be calculated like this:

```
inv.logit<-function(x){exp(x)/(1+exp(x))}
exp_fit<-inv.logit(predict(glm_lambs,newdata=data.frame(MASS=d$MASS
                                                        ,HINDLEG=d$HINDLEG,HORNLEN=d$HORNLEN)))
barW<-mean(exp_fit)
```

Depending on your *R* skills, you might realise that passing the `newdata` argument to the `predict()` function is unnecessary, as `predict()` uses the original data by default. But I did this for a reason. If you wanted to calculate the mean fitness of an hypothetical population where all individuals were just a bit heavier than they really were, you could do this:

```
h<-0.01 # a small numer of KG (or of mm for that matter),
        # relative to the range in the population

bar_W1<-mean(inv.logit(predict(glm_lambs,newdata=data.frame(MASS=d$MASS+h,
                                                        HINDLEG=d$HINDLEG,HORNLEN=d$HORNLEN))))
```

I'll leave it at that. Good luck!

```
# selection gradient for mass
(bar_W1-barW)/h/barW
```

```
## [1] 0.09435126
```

```
# selection gradient for hind leg length
bar_W1<-mean(inv.logit(predict(glm_lambs,newdata=data.frame(MASS=d$MASS,
                                                        HINDLEG=d$HINDLEG+h,HORNLEN=d$HORNLEN))))
(bar_W1-barW)/h/barW
```

```
## [1] 0.003769365
```

```
# selection gradient for horn length
bar_W1<-mean(inv.logit(predict(glm_lambs,newdata=data.frame(MASS=d$MASS,
                                                    HINDLEG=d$HINDLEG,HORNLEN=d$HORNLEN+h))))
(bar_W1-barW)/h/barW
```

```
## [1] -0.00622738
```

I think that you'll find that these compare very well with the OLS estimates we calculated previously.

(15) [Advanced] I claimed that multiple regression is unbiased by corrleations among predictor variables. But I only backed this up in the lecture with a single numerical example. Here is my code again, but with a third predictor variable as a bonus:

```
library(mvtnorm)
P<-matrix(c(1,0.8,-0.2,0.8,1,-0.2,-0.2,-0.2,1),3,3)
b<-c(0.5,-0.5,0.2)
n<-50
x<-rmvnorm(n,rep(0,3),P)
y<-rnorm(n,x%*%b,1)
summary(lm(y~x[,1]+x[,2]+x[,3]))$coefficients
```

```
##                Estimate Std. Error    t value   Pr(>|t|)
## (Intercept)  0.04589998  0.1509850  0.3040035 0.76249617
## x[, 1]       0.64426955  0.2791564  2.3079158 0.02555288
## x[, 2]      -0.80395086  0.2685959 -2.9931614 0.00443052
## x[, 3]       0.31537718  0.1646082  1.9159260 0.06160158
```

Those are not so close to the true values as on my slides. This is because I cranked the sample size way up (on the slides, that is) to make my point in one go. But you might be curious what the situation is for more modest sample size.

Wrap up my example into a simulation that can be run many times, and confirm (or refute) for parameter values of your own choosing whether (a) OLS estimates of linear effects are unbiased (i.e., whether $E[\hat{\beta}] = \beta$), and (b) whether standard errors of OLS regression are rendered incorrect by collinearity (determine the standard error of the estimator is reflective of the standard deviation of the esimates across multiple runs with the same underlying reality).

```
n.sim<-1000
# array with three dimensions: (1) replicates,
# (2) estimate vs SE, and (3) the three predictors
res<-array(dim=c(n.sim,2,3))
for(i in 1:n.sim){
  # code as provided above, except saving results rather than printing to the console
  x<-rmvnorm(n,rep(0,3),P)
  y<-rnorm(n,x%*%b,1)
  co<-summary(lm(y~x[,1]+x[,2]+x[,3]))$coefficients
  res[i,1,]<-co[2:4,1] # estimates
  res[i,2,]<-co[2:4,2] # ses

}

# if unbased, these should be equal
b
```

```
## [1]  0.5 -0.5  0.2
```

```
apply(res[,1,],2,mean)
```

```
## [1]  0.4962368 -0.4951974  0.2059607
```

```
# if SEs are correct, these should be equal
apply(res[,1,],2,sd)
```

```
## [1] 0.2637956 0.2662823 0.1512096
```

```
apply(res[,2,],2,mean)
```

```
## [1] 0.2474220 0.2478731 0.1512431
```

That might have seemed a distraction from selection analysis, but I'm pretty confident that it is counter to a lot of biostatistics teaching, and it is important to be confident (or to know the where the limits are) of the methods you are using, whether for selection analysis or otherwise!

(15) [Advanced] It's not about multivariate selection anymore (in principle; you can build a univariate or a multivairate analysis into your answer for this question), but if you've gotten this far, you're probably finding it pretty fun. Biostatistics orthodoxy has it that regressing a response variable that will have highly non-normal residuals is criminally naive. Actually, OLS requires absolutely no assumption about the distribution of residuals to yield unbiased estimates of linear effects. Do a simulation to determine how well the OLS regression approach of Lande and Arnold recovers selection gradients, when fitness is non-normal (it might be convenient to take advantage of the fact that $\beta = b$ when $E[W] \propto e^{a+bz}$) in order to simulate non-normal fitness residuals, say from a Poisson distribution under GLM assumptions, which being able to fix the true value of β to whatever value you fancy.

```
n.sim<-1000
sim.res<-array(dim=c(n.sim,2)) # track the estimate and its SE
a<-0 # controls mean fitness, and also the degree of non-normality and heteroscedacity
b<-0.5 # beta will be equal to this
n<-40 # a rather small selection study
for(i in 1:n.sim){
  z<-rnorm(n,0,1)
  W<-rpois(n,exp(a+b*z))
  sim.res[i,]<-summary(lm(I(W/mean(W))~z))$coefficients[2,1:2]
}
# bias (unbiased if these are equal)
b
```

```
## [1] 0.5
```

```
mean(sim.res[,1])
```

```
## [1] 0.4903994
```

```
# adequacy of SEs (good if these are equal)
sd(sim.res[,1])
```

```
## [1] 0.1632124
```

```
mean(sim.res[,2])
```

```
## [1] 0.1564985
```

The calculation of SEs for OLS regression also makes no assumption about normality of residuals, but does require that they have equal variance across values of the predictor (z in our case). These are not equal under our poisson model. But a broken assumption does not a catastrophe make, and OLS is not massively subject to catastrophe.

As a last-ditch effort to save the notion about assuming normal residuals, people confront me with the fact that normal residuals are required for null hypothesis statistical testing and for constructing confidence

intervals. This is strictly true, but from the central limit theorem only matters at very small sample size. For the simulation as I've done it with a very modest sample size, how to basic wald-type confidence intervals hold up? This requires working out how often the method for constructing the confidence interval contains the true value:

```
est<-sim.res[,1]
se<-sim.res[,2]
table(b>(est-1.96*se) & b<(est+1.96*se))/n.sim
```

```
##
## FALSE  TRUE
## 0.056 0.944
```

If the devil offers you a deal that every statistical analysis you do for the rest of your life will be this wrong, but no more wrong, I suggest you take it.

Elaborations

(16) Calculate the directional selection differentials and gradients for the two episodes I used (first year viability selection, and subsequent lifetime selection of survivors), and use the Arnold-Wade-Kalisz mechanics for the partition of total selection into episodes to combine them to recover total selection.

```
# from above, S and beta for viability episode
S_mass
```

```
## [1] 0.3917634
```

```
beta_mass
```

```
## [1] 0.08186105
```

```
# S and beta for subsequent lifetime selection among survivors
s<-subset(d,d$SURV==1)
S_mass_b<-weighted.mean(s$MASS,s$LRS)-mean(s$MASS)
S_mass_b
```

```
## [1] 0.4851291
```

```
beta_mass_b<-S_mass_b/var(s$MASS)
beta_mass_b
```

```
## [1] 0.1107441
```

... now the additive partition of S and the weighted sum for β

```
# the additive partition of S
S_mass+S_mass_b
```

```
## [1] 0.8768925
```

```
# the weighted partition of beta
beta_mass + var(s$MASS)/var(d$MASS)*beta_mass_b
```

```
## [1] 0.1832313
```

Here arre the total gradients, which will serve as a check that the right system of combining S and β over episodes was achieved:

```
weighted.mean(d$MASS,d$LRS)-mean(d$MASS)
```

```
## [1] 0.8768925
```

```
(weighted.mean(d$MASS,d$LRS)-mean(d$MASS))/var(d$MASS)
```

```
## [1] 0.1832313
```

(17a) Estimate the matrix of quadratic (including correlational) viability selection gradients for mass, leg length, and horn length in Soay ewe lambs. (You may or may not have already done this for question 12, though you may want to re-do it with variance standardisation, so it will match the lecture.)

```
vstand<-function(x){(x-mean(x))/sd(x)}
d$MASS_vstand<-vstand(d$MASS)
d$HINDLEG_vstand<-vstand(d$HINDLEG)
d$HORNLEN_vstand<-vstand(d$HORNLEN)
```

```
m2<-lm(w_s~MASS_vstand+HINDLEG_vstand+HORNLEN_vstand
          +I(0.5*MASS_vstand^2)+I(0.5*HINDLEG_vstand^2)+I(0.5*HORNLEN_vstand^2)
          +I(MASS_vstand*HINDLEG_vstand)+I(MASS_vstand*HORNLEN_vstand)
                +I(HINDLEG_vstand*HORNLEN_vstand),data=d)

# directional gradients
summary(m2)$coefficients[2:4,]
```

```
##                  Estimate Std. Error   t value    Pr(>|t|)
## MASS_vstand     0.1658371 0.06648779  2.494250 0.013420501
## HINDLEG_vstand  0.0766935 0.06518176  1.176610 0.240729089
## HORNLEN_vstand -0.1241003 0.04592976 -2.701958 0.007476474
```

```
# quadratic gradients
summary(m2)$coefficients[5:7,]
```

```
##                              Estimate Std. Error   t value   Pr(>|t|)
## I(0.5 * MASS_vstand^2)    -0.34991155 0.18015389 -1.942292 0.05348521
## I(0.5 * HINDLEG_vstand^2) -0.33042563 0.16149190 -2.046082 0.04203760
## I(0.5 * HORNLEN_vstand^2) -0.05245746 0.04847223 -1.082217 0.28044005
```

```
# correlational gradients
summary(m2)$coefficients[8:10,]
```

```
##                                      Estimate Std. Error    t value
## I(MASS_vstand * HINDLEG_vstand)     0.29552334 0.14598950  2.0242781
## I(MASS_vstand * HORNLEN_vstand)     0.15594066 0.09017127  1.7293829
## I(HINDLEG_vstand * HORNLEN_vstand) -0.06880514 0.08459723 -0.8133261
##                                       Pr(>|t|)
## I(MASS_vstand * HINDLEG_vstand)     0.04425177
## I(MASS_vstand * HORNLEN_vstand)     0.08526073
## I(HINDLEG_vstand * HORNLEN_vstand) 0.41698401
```

(17b) [tricky] Visualise the two major axes of quadratic selection. It is quite a bit of work to do this in R. First, we need to construct the gamma matrix:

```
gamma<-diag(coef(m2)[5:7])
gamma[lower.tri(gamma)]<-coef(m2)[8:10]
gamma[upper.tri(gamma)]<-coef(m2)[8:10]
gamma
```

```
##            [,1]        [,2]        [,3]
## [1,] -0.3499115  0.29552334  0.15594066
```

```
## [2,]  0.2955233 -0.33042563 -0.06880514
## [3,]  0.1559407 -0.06880514 -0.05245746
```

Now, we need to get the eigenvalues:

```
e<-eigen(gamma)
e
```

```
## eigen() decomposition
## $values
## [1]  0.03159236 -0.08723544 -0.67715156
##
## $vectors
##            [,1]       [,2]       [,3]
## [1,] -0.5523408 -0.4338085  0.7118496
## [2,] -0.3033101 -0.6908158 -0.6563356
## [3,] -0.7764809  0.5784321 -0.2499873
```

`eigen()` orders according to most positive to most negative eigenvalues. We want to order according to absolute values of eigenvalues

```
o<-order(abs(e$values),decreasing=TRUE)
o
```

```
## [1] 3 2 1
```

```
e$values<-e$values[o]
e$vectors<-e$vectors[,o]
e
```

```
## eigen() decomposition
## $values
## [1] -0.67715156 -0.08723544  0.03159236
##
## $vectors
##            [,1]       [,2]       [,3]
## [1,]  0.7118496 -0.4338085 -0.5523408
## [2,] -0.6563356 -0.6908158 -0.3033101
## [3,] -0.2499873  0.5784321 -0.7764809
```

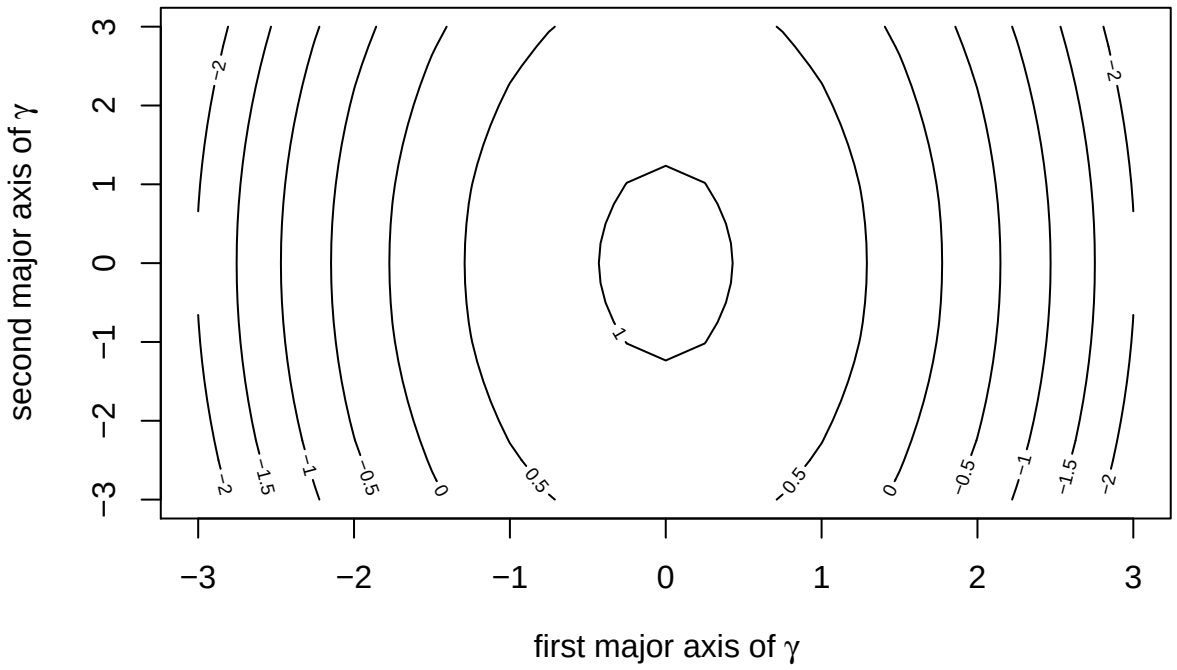
Now, we need a range of values over which to plot contours. Lets go from -3 to +3 on both axes

```
x<-seq(-3,3,length.out=25)
x<-expand.grid(ga1=x,ga2=x)
```

Now we need the quadratic component of the fitness surface, calculated for all these combinations of values

```
x$w<-coef(m2)[1] + e$values[1]*0.5*x$ga1^2 + e$values[2]*0.5*x$ga2^2
```

```
contour(matrix(x$w,25,25),xaxt='n',yaxt='n'
        ,xlab=expression(paste("first major axis of ",gamma))
        ,ylab=expression(paste("second major axis of ",gamma)))
axis(side=1,seq(0,1,length.out=7),-3:3)
axis(side=2,seq(0,1,length.out=7),-3:3)
```



You might want to check out `help(contour)`. It is particularly important to know which way the matrix is being filled (by columns, by default, and that this maps on to the use of `expand.grid()` to set up the combinations of predictor variable values, and the intended labels for the x and y axes. It is easy to get your axes reversed, and it could happen at any of these three stages.)

(18) Say an ecophysiologist (I fear probably not a terribly competent one, but we'll run with it) told you that mass is almost entirely causally determined by skeletal size (for which we have a good proxy in hind leg length), and that skeletal size and mass are both likely to have direct effects on fitness. This might suggest a path model to you wherein *hind leg* → *mass*, *hind leg* → *survival* & *mass* → *survival*. Calculate the direct and extended sense viability selection gradients of hind leg length and mass, and compare these to the direct selection gradients (Lande's β).

This requires two regression models to calculate the parameters of the path model. The first one is of the two effects on fitness:

```
path1<-lm(w_s~HINDLEG_vstand+MASS_vstand,data=d)
summary(path1)$coefficients
```

```
##                  Estimate Std. Error    t value     Pr(>|t|)
## (Intercept)    1.00000000 0.03185444 31.3927947 2.963847e-81
## HINDLEG_vstand 0.03443935 0.06287576  0.5477364 5.844546e-01
## MASS_vstand    0.15025792 0.06287576  2.3897589 1.774345e-02
```

and the next is of the effect of leg length on mass

```
path2<-lm(MASS_vstand~HINDLEG_vstand,data=d)
summary(path2)$coefficients
```

```
##                    Estimate Std. Error      t value     Pr(>|t|)
## (Intercept)    1.884339e-16 0.03487751 5.402733e-15 1.000000e+00
## HINDLEG_vstand 8.614640e-01 0.03495967 2.464165e+01 5.103391e-64
```

The effects in the first model are the direct effects on fitness, or in other words, β

```
beta<-coef(path1)[2:3]
beta
```

```
## HINDLEG_vstand    MASS_vstand
##     0.03443935     0.15025792
```

The developmental entanglements among traits are simple in this model, being only the effect of skeletal size on mass:

```
b<-matrix(c(0,coef(path2)[2],0,0),2,2,byrow=TRUE)
b
```

```
##      [,1]     [,2]
## [1,]    0 0.861464
## [2,]    0 0.000000
```

The total effects of traits on one another are

```
phi<-solve(diag(2)-b)
phi
```

```
##      [,1]     [,2]
## [1,]    1 0.861464
## [2,]    0 1.000000
```

And finally, the extended selection gradients for structural size and mass are

```
phi%*%beta
```

```
##           [,1]
## [1,] 0.1638811
## [2,] 0.1502579
```

We can check that we got the linear algebra all organised right by checking against the solution according to path rules. The only need for path analysis is to note that the extended sense selection gradient is its direct effet on fitness, plus the product of its effect on mass and mass' direct effect on fitness

```
beta[1]+b[1,2]*beta[2]
```

```
## HINDLEG_vstand
##      0.1638811
```

which agrees.

(19) The file `soay_ram_lamb_selection_data.csv` contains analogous data for males – what fun! Calculate directional viability selection gradients for mass, hind leg length, and horn length. Compare these with the corresponding selection gradients for females. Consider how these selection gradients are represented under Cheng and Houle's formula for expressing sex-specific selection as concondant and antagonsitic selection gradient vectors.

Read in the data

```
dm<-read.table("./soay_ram_lamb_selection_data.csv",header=TRUE,sep=',')
```

Fit a multivariate selection model. It need not be a full quadratic model, and also the standardisation doesn't necessarily have to be to the variance. However, we have a fitted model `m2` in memory for the females, so for the purpose of this illustrative example, I'm just making the male model analogous to that female model.

```
vstand<-function(x){(x-mean(x))/sd(x)}
dm$MASS_vstand<-vstand(dm$MASS)
dm$HINDLEG_vstand<-vstand(dm$HINDLEG)
dm$HORNLEN_vstand<-vstand(dm$HORNLEN)
dm$w_s<-dm$SURV/mean(dm$SURV)
```

```
m2m<-lm(w_s~MASS_vstand+HINDLEG_vstand+HORNLEN_vstand
          +I(0.5*MASS_vstand^2)+I(0.5*HINDLEG_vstand^2)+I(0.5*HORNLEN_vstand^2)
          +I(MASS_vstand*HINDLEG_vstand)+I(MASS_vstand*HORNLEN_vstand)
                +I(HINDLEG_vstand*HORNLEN_vstand),data=dm)

# directional gradients
summary(m2m)$coefficients[2:4,]
```

```
##                   Estimate Std. Error   t value     Pr(>|t|)
## MASS_vstand    -0.08459836 0.02778804 -3.044416 2.429808e-03
## HINDLEG_vstand  0.13414362 0.03053551  4.393037 1.313912e-05
## HORNLEN_vstand -0.05174994 0.03037205 -1.703867 8.890535e-02
```

```
# quadratic gradients
summary(m2m)$coefficients[5:7,]
```

```
##                              Estimate Std. Error    t value   Pr(>|t|)
## I(0.5 * MASS_vstand^2)    -0.13095372 0.06244628 -2.0970621 0.03639123
## I(0.5 * HINDLEG_vstand^2) -0.02273767 0.04518461 -0.5032172 0.61498970
## I(0.5 * HORNLEN_vstand^2)  0.04211005 0.05785524  0.7278520 0.46697810
```

```
# correlational gradients
summary(m2m)$coefficients[8:10,]
```

```
##                                      Estimate Std. Error    t value
## I(MASS_vstand * HINDLEG_vstand)    0.022778631 0.03125171 0.72887623
## I(MASS_vstand * HORNLEN_vstand)    0.023050685 0.03157415 0.73004924
## I(HINDLEG_vstand * HORNLEN_vstand) 0.002995542 0.03446870 0.08690615
##                                       Pr(>|t|)
## I(MASS_vstand * HINDLEG_vstand)    0.4663517
## I(MASS_vstand * HORNLEN_vstand)    0.4656349
## I(HINDLEG_vstand * HORNLEN_vstand) 0.9307741
```

So the selection gradient estimtes with SEs (both from full quadratic models, but only considerting directional selection) are:

```
beta_m<-coef(m2m)[2:4]
beta_f<-coef(m2)[2:4]
beta_m
```

```
##    MASS_vstand HINDLEG_vstand HORNLEN_vstand
##    -0.08459836     0.13414362    -0.05174994
```

```
beta_f
```

```
##    MASS_vstand HINDLEG_vstand HORNLEN_vstand
##      0.1658371      0.0766935     -0.1241003
```

The transformation of these into concordant and antagonsitic selection vectors for the three traits is (from lecture 4)

$$\boldsymbol{\beta}_{ac} = \begin{bmatrix} \boldsymbol{\beta}_a \\ \boldsymbol{\beta}_c \end{bmatrix} = \mathbf{Q}_{mf \to ac} \begin{bmatrix} \boldsymbol{\beta}_m \\ \boldsymbol{\beta}_f \end{bmatrix}$$

where

$$= \mathbf{Q}_{mf \to ac} \begin{bmatrix} \mathbf{I} & \mathbf{I} \\ \mathbf{I} & -\mathbf{I} \end{bmatrix}$$

so, in R:

```
# make Q matrix and check it is right
Q<-diag(6)
Q[1:3,4:6]<-Q[4:6,1:3]<-diag(3)
Q[4:6,4:6]<--diag(3)
Q
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6]
## [1,]    1    0    0    1    0    0
## [2,]    0    1    0    0    1    0
## [3,]    0    0    1    0    0    1
## [4,]    1    0    0   -1    0    0
## [5,]    0    1    0    0   -1    0
## [6,]    0    0    1    0    0   -1
```

```
# transform sex-specific selection gradient matrices
Q %*% matrix(c(beta_m,beta_f),6,1)
```

```
##             [,1]
## [1,]  0.08123878
## [2,]  0.21083712
## [3,] -0.17585024
## [4,] -0.25043551
## [5,]  0.05745012
## [6,]  0.07235035
```

So, direct selection on mass has a very large antagonistic component, while selection is generally concordant on the other two trraits.

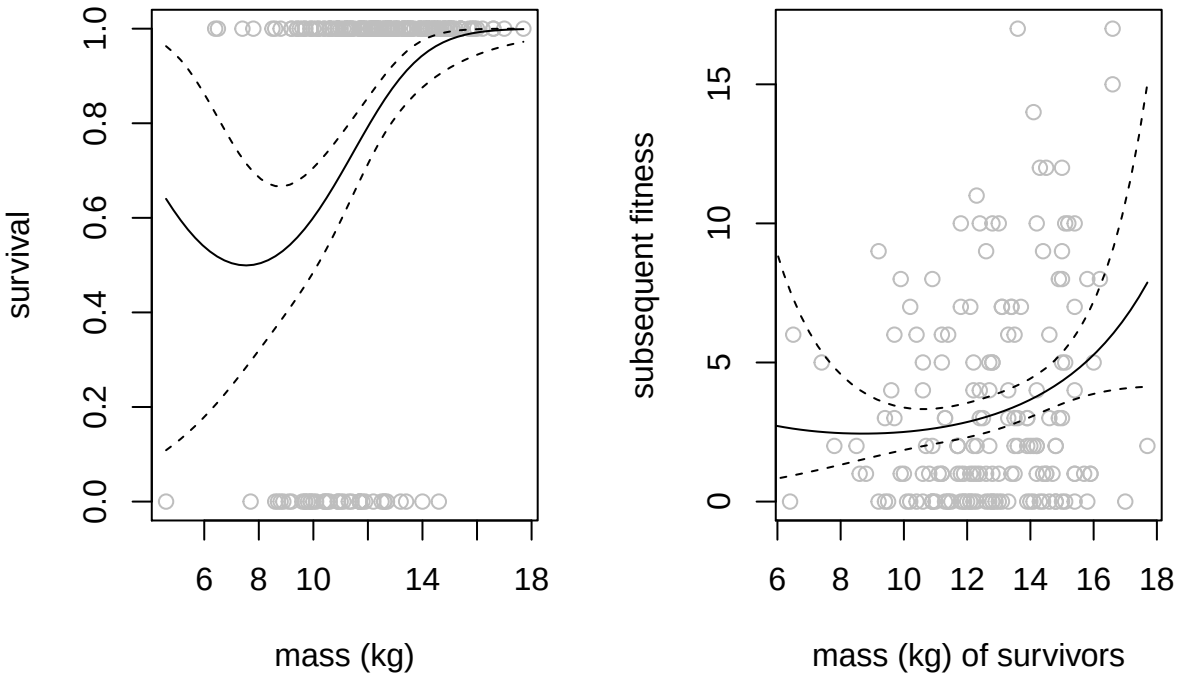
(20) [Advanced] Fit a GLM of the relationship between lamb August mass of Soay ewes that survive their first winter. In combination with a GLM of viability selection (maybe simplify the one from question 14 down to using just mass as a predictor), make a model of lifetime selection of August lamb mass that is sensitive to the statistial distributions of survival and subsequent total reproductive success.

Fit the two GLMs (I think a quasipoisson GLM is sensible, but this shouldn't be too critical):

```
glm_s<-glm(SURV~MASS+I(MASS^2),family='binomial',data=d)
glm_lrs<-glm(LRS~MASS+I(MASS^2),family='quasipoisson',data=s)
```

Visualise the two GLMs:

```
par(mfrow=c(1,2))
x<-seq(min(d$MASS),max(d$MASS),length.out=50)
plot(d$MASS,d$SURV,col="gray",xlab="mass (kg)",ylab="survival")
p1<-predict(glm_s,data.frame(MASS=x),se.fit=TRUE)
lines(x,inv.logit(p1$fit))
lines(x,inv.logit(p1$fit+1.96*p1$se.fit),lty="dashed")
lines(x,inv.logit(p1$fit-1.96*p1$se.fit),lty="dashed")
plot(s$MASS,s$LRS,col="gray",xlab="mass (kg) of survivors",ylab="subsequent fitness")
p2<-predict(glm_lrs,data.frame(MASS=x),se.fit=TRUE)
lines(x,exp(p2$fit))
lines(x,exp(p2$fit+1.96*p2$se.fit),lty="dashed")
lines(x,exp(p2$fit-1.96*p2$se.fit),lty="dashed")
```



Here is a function that returns expected values of fitness components and total fitness, for any set of individual phenotypes, assuming that the fitted GLM objects `glm_s` and `glm_lrs` are in memory:

```
mean_fitnesses<-function(z){
  W1<-inv.logit(predict(glm_s,data.frame(MASS=z)))
  W2<-exp(predict(glm_lrs,data.frame(MASS=z)))
  Wt<-W1*W2
  return(c(mean(W1),mean(W2),mean(Wt)))
}
```

This function makes it easy to apply the average derivative method

```
barW<-mean_fitnesses(d$MASS)
h<-0.01
barW1<-mean_fitnesses(d$MASS+h)
beta<-(barW1-barW)/h/barW
beta
```

```
## [1] 0.08048248 0.11987324 0.20125248
```

The selection gradient for the second episode, calculated this way, has a slightly nuanced interpretation. The selection coefficients that we estimated before (directly from the data) were based only on data from survivors of the first episode of (viability) selection. While the function relating mass to subsequent fitness that is used herre is also necerssarily estimated only from survivors, we took the gradient over the full distribution of mass, before the first episode of viability selection. Consequently, these selection gradients are additive.

Making estimates (these estimates of β) are great, but it is a bit unsatisfying unless you can put some kind of statements about statistical uncertainty to them. I would bootstrap such an analysis like this:

```
n.boot<-1000
boot.res<-array(dim=c(n.boot,3))
h<-0.01

for(i in 1:n.boot){
  boot.d<-d[sample(1:(dim(d)[1]),dim(d)[1],replace=TRUE),]
```

```
  glm_s<-glm(SURV~MASS+I(MASS^2),family='binomial',data=boot.d)
  boot.s<-subset(boot.d,boot.d$SURV==1)
  glm_lrs<-glm(LRS~MASS+I(MASS^2),family='quasipoisson',data=boot.s)
  barW<-mean_fitnesses(boot.d$MASS)
  barW1<-mean_fitnesses(boot.d$MASS+h)
  boot.res[i,]<-(barW1-barW)/h/barW
}
```

The standard deviation of the bootstrap estimates of β serve as the estimates of the standard errors of the selection gradients:

```
beta
```

```
## [1] 0.08048248 0.11987324 0.20125248
```

```
apply(boot.res,2,sd)
```

```
## [1] 0.01611793 0.04667534 0.04918917
```