Additional documentation for GSG

Michael B. Morrissey[1] and Krzysztof Sakrejda[2]

December 22, 2013

[1]michael.morrissey@st-andrews.ac.uk
[2]sakrejda@cns.umass.edu

Contents

1 Selection gradients and fitness functions for human birth weight and gestation length via variation in neonatal survival

The tensor product smooth-based generalized additive model in Morrissey and Sakrejda (2013) was fitted by:

```
library(mgcv)
data(humanNeonatal)
neonatalGam <- gam(nns˜te(bw,gest), family=’binomial’, data=humanNeonatal)
```

We then used the function `gam.gradients()` to obtain selection gradients

```
> library(gsg)
> gradientsGam <- gam.gradients(neonatalGam, phenotype=c("bw","gest"),
+              n.boot=1000, standardize=TRUE)
Calculating bootstrap standard errors...

      ... estimated completion at  2012-06-10 16:19:03 ...done.
>
> round(gradientsGam,4)
          estimates     SE P.value
B-bw         0.0223 0.0034   0.000
B-gest       0.0037 0.0031   0.242
G-bw        -0.0350 0.0048   0.000
G-gest      -0.0087 0.0025   0.000
G-bw-gest   -0.0042 0.0037   0.300
```

The computation with 1000 bootstrap replicates took approximately 1.9 hours using a personal computer with an Intel Core 2 processor at 1.8 GHz. The same computation required approximately 7.5 minutes on an Intel i7 at 4.2 GHz using 4 cores.

2 Plotting a fitness landscape

The bivariate fitness landscape in Morrissey and Sakrejda (2013) was obtained by:

```
neonatal.fl<-fitness.landscape(mod= neonatalGam,
         phenotype=c("bw","gest"),plt.density=10,PI.method=’n’)
```

and the plot was made similarly to:

```
p<-matrix(neonatal.fl$Wbar,10,10,byrow=TRUE)
par(mar=c(5.5,6,1,1),oma=rep(1,4),las=1,cex.lab=1.2)
```

```
contour(t(p),xaxt=’n’,yaxt=’n’,xlab="Mean birth mass (kg)",ylab="")
axis(at=seq(0,1,length.out=10),
        round(unique(neonatal.fl$points[,1]),2),side=1)
axis(at=seq(0,1,length.out=10),
        round(unique(neonatal.fl$points[,2]),2),side=2)
par(las=0)
mtext(side=2,outer=TRUE,line=-1.5,
        "Mean gestation length (days)",cex=1.2)
```

3 The Lande-Arnold selection analysis as a special case

A quadratic approximation of the bivariate human neonatal fitness function can be obtained by:

```
neonatalQuadratic <- gam(nns~bw+gest+I(bw^2)+
               I(gest^2)+I(bw*gest), family=’gaussian’,
               data=humanNeonatal)
```

Obtaining the first and second order partial derivatives of this function is an implementation of the Lande and Arnold (1983) selection analysis as a special case of the general formulation described in Morrissey and Sakrejda (2013):

```
> gradientsQuadratic <- gam.gradients(neonatalQuadratic,
+             phenotype=c("bw","gest"),
+             n.boot=1000, standardize=TRUE)
Calculating bootstrap standard errors...

      ... estimated completion at  2012-06-10 17:00:13 ...done.
>
> round(gradientsQuadratic,4)
          estimates     SE P.value
B-bw         0.0292 0.0040   0.000
B-gest       0.0045 0.0035   0.198
G-bw        -0.0599 0.0059   0.000
G-gest      -0.0171 0.0049   0.000
G-bw-gest   -0.0102 0.0042   0.012
```

Note that standardizations necessary for the Lande and Arnold (1983) analysis (mean standardization of traits and analysis of fitness on the relative scale, scaling of 0.5 for the diagonal quadratic coefficients; Stinchcombe et al. 2008) are intrinsic to the calculations implemented in `gam.gradients`:

```
humanNeonatal$st.bw <- (humanNeonatal$bw-mean(humanNeonatal$bw))/
                        sd(humanNeonatal$bw)
humanNeonatal$st.gest <- (humanNeonatal$gest-mean(humanNeonatal$gest))/
                        sd(humanNeonatal$gest)
humanNeonatal$w<-humanNeonatal$nns/mean(humanNeonatal$nns)
neonatalQuadraticStandardized <- gam(w~ st.bw + st.gest +I(0.5* st.bw^2)
                        +I(0.5*st.gest^2)+I(st.bw*st.gest), family='gaussian',
                        data=humanNeonatal)
gradientsQuadraticS <- gam.gradients(neonatalQuadraticStandardized,
                       phenotype=c("st.bw","st.gest"),
                       n.boot=1000, standardize=TRUE)
```

This produces the same selection gradients estimates. Differences in the standard errors are due to MC error.

```
> round(gradientsQuadraticS,4)
                estimates     SE P.value
B-st.bw            0.0292 0.0038   0.000
B-st.gest          0.0045 0.0035   0.190
G-st.bw           -0.0599 0.0063   0.000
G-st.gest         -0.0171 0.0048   0.000
G-st.bw-st.gest   -0.0102 0.0042   0.018
```

4 Compromises between model flexibility and simplicity

As acknowledged in the discussion of Morrissey and Sakrejda (2013), it will not always be sensible to fit fully flexible smooth terms for characterizing multivariate fitness functions. The large neonatal survival databased allowed the bivariate tensor product smooth to be fitted, but such data are often not available in evolutionary quantitative genetic studies of wild populations. Slightly less flexible models may often be sensible, and can be handled in the analytical framework supported by the R package GSG. A generally useful approach may be to model fitness as semi-parametric smooth functions of each variable, while handling interactions parametrically. This fitness function could be applied to the analysis of the human neonatal data via:

```
neonatalLessFlexible<-gam(nns~s(bw)+s(gest)+bw:gest,
                family=’binomial’,data=humanNeonatal)
```

Analysis based on this somewhat less flexible characterization of the fitness function proceeds similarly, and provides very similar results:

```
> gradientsLessFlexible<-gam.gradients(neonatalLessFlexible,
+                          phenotype=c("bw","gest"),
+                          n.boot=1000, standardize=TRUE)
Calculating bootstrap standard errors...

      ... estimated completion at  2012-06-11 09:20:08 ...done.
> round(gradientsLessFlexible,4)
          estimates     SE P.value
B-bw         0.0217 0.0038   0.000
B-gest       0.0033 0.0033   0.346
G-bw        -0.0339 0.0063   0.000
G-gest      -0.0184 0.0045   0.000
G-bw-gest   -0.0019 0.0034   0.542
```

This more constrained model may in fact have some interpretive benefits, for example, the lack of statistical support for the interaction between birth weight and gestation length in the fitness function compliments the estimate of the small (and also statistically unsupported) off-diagonal element of the matrix of quadratic selection coefficients (see above and Morrissey and Sakrejda 2013):

```
> summary(neonatalLessFlexible)

Family: binomial
Link function: logit

Formula:
nns ~ s(bw) + s(gest) + bw:gest

Parametric coefficients:
              Estimate Std. Error z value Pr(>|z|)
(Intercept)  3.7033796  4.5862541   0.807    0.419
bw:gest     -0.0005008  0.0051294  -0.098    0.922

Approximate significance of smooth terms:
          edf Ref.df Chi.sq  p-value
s(bw)   3.861  4.843 113.24  < 2e-16 ***
s(gest) 5.073  6.090  30.74 3.09e-05 ***
```

```
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1   1

R-sq.(adj) =  0.235   Deviance explained = 22.7%
UBRE score = -0.67517  Scale est. = 1         n = 7036
```

5 Notes about algorithms for calculating standard errors and/or p-values

The parametric bootstrap, as applied in Morrissey and Sakrejda (2013) is the default method for obtaining coefficients of selection gradients and prediction intervals fitness landscapes, in each function in GSG. Alternative algorithms include case bootstrapping, simulation from an approximation to the posterior distribution of the gam model parameters, and a permutation test (P-values only). The two bootstrap algorithms, and the posterior simulations, allow the smoothing parameters to be fixed across replicates, or re-estimated. By default, they are fixed following Schluter (1988).

6 A brief example with a Poisson fitness response

Fitness data are often counts, and so reasonably modelled as Poisson variables. Implementing the methods described in Morrissey and Sakrejda (2013) using GSG is straightforward for Poisson or other fitness distributions is straightforward. The functions in GSG that extract data from a fitted `gam` object rely on prediction on the data scale, and so analysis based on different assumed distributions of fitness simply require fitting a model with a different error structure.

The example code below simulates a Poisson fitness response as a function of a single trait, and shows the implementation of an analysis to obtain the associated selection gradient:

```
> n<-200
> z<-rnorm(n,0,1)
> W<-rpois(n,exp(1+z-0.5*z^2))
> simPoisData<-as.data.frame(list(z=z,W=W))
>
> simPoisGam<-gam(W~s(z),family=’poisson’,data=simPoisData)
>
> gradientsPoisSim<-gam.gradients(simPoisGam,phenotype="z")
Calculating bootstrap standard errors...[1] 100

      ... estimated completion at  2012-06-11 09:30:52 ...done.
>
> round(gradientsPoisSim,4)
    estimates     SE P.value
B-z    0.4423 0.0642   0.000
G-z   -0.2068 0.0852   0.034
```

7 Direct calculation of selection differentials

Selection differentials are defined most simply as the change in the central moments of the phenotypic distribution due to selection (Endler, 1986; Lande and Arnold, 1983). Generally, these can be calculated as the difference between the means, variances, and covariances, weighted by fitness, and the unweighted moments. These are calculated using `moments.differentials()` in the R package GSG

```
> humanDifferentials<-moments.differentials(
+         z=humanNeonatal[,c("bw","gest")],
+         W=humanNeonatal$nns,n.boot=1000,standardized=TRUE)
>
> round(humanDifferentials,4)
      Coefficient     SE P-value
S 1        0.0667 0.0055       0
S 2        0.0612 0.0056       0
C 1       -0.2057 0.0153       0
C 2       -0.2160 0.0183       0
C 1,2     -0.1919 0.0157       0
```

8 Lasso and ridge regression selection analysis

Selection gradients were obtained from the regularised regression analyses in Morrissey (2013) by tricking `gam.gradients()` into doing the analysis. First the regression analyses were fitted; using the lasso as an example:

```
library(glmnet)
data(SoayLambs)

phen<-c("WEIGHT","HINDLEG","HORNLEN","lnKeds")
covars<-SoayLambs[, phen]
for(i in 1:4){
  for(j in 1:i){
    covars<-cbind(covars,covars[,phen[i]]*covars[,phen[j]])
    names(covars)[length(names(covars))]<-paste(phen[i],phen[j],sep="")
  }
}


lamb.lasso<-cv.glmnet(x=as.matrix(covars), y=
     SoayLambs$W, family=’binomial’,alpha=1)
```

The coefficients of the fitted lasso model are thus:

```
> predict(lamb.lasso,type="coefficients",s="lambda.min")
15 x 1 sparse Matrix of class "dgCMatrix"
                        1
(Intercept)     1.6051944
WEIGHT          1.0970974
HINDLEG         0.2661741
HORNLEN        -0.4738859
lnKeds         -0.2270427
WEIGHTWEIGHT    0.1396266
HINDLEGWEIGHT   .
HINDLEGHINDLEG  .
HORNLENWEIGHT   0.0687889
HORNLENHINDLEG  .
HORNLENHORNLEN  .
lnKedsWEIGHT    .
lnKedsHINDLEG  -0.2111347
lnKedsHORNLEN   .
lnKedslnKeds    .
>
```

These can be forced into a gam object, and then the gradients are obtained using `gam.gradients()`:

```
dummy.gam<-gam(W~WEIGHT+HINDLEG+HORNLEN+lnKeds
   +I(WEIGHT^2)
   +I(WEIGHT*HINDLEG)  +I(HINDLEG^2)
   +I(WEIGHT*HORNLEN)  +I(HINDLEG*HORNLEN)  +I(HORNLEN^2)
   +I(WEIGHT*lnKeds)   +I(HINDLEG*lnKeds)   +I(HORNLEN*lnKeds) + I(lnKeds^2),
           family=’binomial’,data= SoayLambs)

predict(lamb.lasso,type="coefficients",s="lambda.min")

lasso.coefs<-as.numeric(predict(lamb.lasso,type="coefficients",s="lambda.min"))
dummy.gam$coefficients<-lasso.coefs

lasso.grads<-gam.gradients(mod=dummy.gam,phenotype=phen,se.method=’n’)
```

The lasso-based selection gradients are thus:

```
> lasso.grads
                     estimates SE P.value
B-WEIGHT           0.161052875 NA      NA
B-HINDLEG          0.039538491 NA      NA
B-HORNLEN         -0.086004182 NA      NA
B-lnKeds          -0.022464858 NA      NA
G-WEIGHT          -0.046488274 NA      NA
G-HINDLEG         -0.007482505 NA      NA
G-HORNLEN         -0.017072158 NA      NA
G-lnKeds          -0.005697433 NA      NA
G-WEIGHT-HINDLEG  -0.020150751 NA      NA
G-WEIGHT-HORNLEN   0.049823337 NA      NA
G-HINDLEG-HORNLEN  0.008351592 NA      NA
G-WEIGHT-lnKeds    0.020535502 NA      NA
G-HINDLEG-lnKeds  -0.031363207 NA      NA
G-HORNLEN-lnKeds  -0.007713071 NA      NA
```

Obtaining standard errors and P-values for such an analysis does not seem meaningful, as the shrinkage and variable selection inherent in the lasso (or elastic net regression, generally) to some extent generates parameters that reflect both the pattern in the data and the extent to which it is statistically supported.

9 Generalised projection-pursuit regression and selection gradients

Characterisation of a fitness landscape might proceed as above:

```
data(SoayLambs)
phen<-c("WEIGHT","HINDLEG","HORNLEN","lnKeds")
fit.land<-gppr(y="W",xterms=phen,
              data=SoayLambs,family=’binomial’)

grads<-gppr.gradients(mod=fit.land,
              phenotype=phen,
              family=’binomial’)
```

In which case the gradients produced are

```
> grads$ests
                      estimates          SE P.value
B-WEIGHT           1.942585e-01 0.063735725   0.002
B-HINDLEG          3.779547e-02 0.059054385   0.494
B-HORNLEN         -1.212769e-01 0.044498700   0.006
B-lnKeds          -3.235531e-02 0.031340850   0.284
G-WEIGHT          -8.071137e-02 0.054259581   0.012
G-HINDLEG          4.556036e-05 0.015213039   0.660
G-HORNLEN         -2.682024e-02 0.025293852   0.018
G-lnKeds           5.079316e-04 0.004820774   0.930
G-WEIGHT-HINDLEG  -1.956484e-02 0.024748309   0.496
G-WEIGHT-HORNLEN   5.039030e-02 0.028296653   0.008
G-HINDLEG-HORNLEN  8.371374e-03 0.018093772   0.502
G-WEIGHT-lnKeds    1.344184e-02 0.014885079   0.292
G-HINDLEG-lnKeds  -4.058068e-05 0.006674169   0.866
G-HORNLEN-lnKeds  -1.045944e-02 0.010646091   0.276
>
```

One might wish to obtain the selection gradients associated with the axes of phenotype of the gppr analysis. This could be done by re-fitting a gam with the same type of regression function to rotated data:

```
SoayLambs$SelTerm<-as.matrix(SoayLambs[,phen]) %*% as.matrix(fit.land$alpha)

new.mod<-gam(W~s(SelTerm,bs="cr"),data=SoayLambs,family=’binomial’)

grads2<-gam.gradients(mod=new.mod,phenotype="SelTerm",standardized=TRUE)
```

This yields the gradient estimates of selection along the axis defined by the gppr as

```
> grads2$ests
            estimates         SE P.value
B-SelTerm  0.18616108 0.03970303   0.000
G-SelTerm -0.05821734 0.10574565   0.218
>
```

The SEs and P-values should be taken with a grain of salt. Since the gppr analysis has specifically sought to find an axis that explains fitness variation, statistical inference of selection focusing only on that direction, and not accounting for all the other directions that were not chosen, is inappropriate. The P-values should thus be thought of as requiring correction for multiple testing, although just how many tests (i.e., of phenotypic directions) one should think of the gppr analysis as having conducted, I don't know.

It seems that it should be instructive, at least, to consider what variance in expected fitness would have been apparently explained under an hypothesis of no selection. Although I used this approach in Morrissey (2013), I do not specifically want to promote it at present as a "canned solution", so it is not specifically implemented in any function in `gsg`. This approach is pretty easily implemented, though:

```
n.perm<-1000
varWperm<-array(dim=1000)
for(x in 1:n.perm){
  SoayLambs$permutedW<-SoayLambs$W[sample(1:length(SoayLambs$W),
                       length(SoayLambs$W),replace=FALSE)]
  perm.mod<-gppr(y="permutedW",xterms=phen,
                data=SoayLambs,family='binomial')
  varWperm[x]<-var(inv.logit(predict(perm.mod,type="raw")))
}
```

The variance in expected absolute fitness (survival probability) from the fitted model is

```
> var(inv.logit(predict(fit.land,type="raw")))
[1] 0.02917478
>
```

which is very much in the tail of our null distribution

```
> table(var(inv.logit(predict(fit.land,type="raw")))>varWperm)/n.perm

TRUE
   1
>
> quantile(varWperm,probs=c(0.025,0.25,0.5,0.75,0.975))
        2.5%          25%          50%          75%        97.5%
0.0004020845 0.0016562070 0.0027748300 0.0045031314 0.0097702922
>
```

References

Endler, J. A., 1986. Natural selection in the wild. Princeton University Press.

Lande, R. and S. J. Arnold, 1983. The measurement of selection on correlated characters. Evolution 37:1210–1226.

Morrissey, M. B., 2013. In search of the best methods for multivariate selection analysis. in preparation for submission to Methods in Ecology and Evolution .

Morrissey, M. B. and K. Sakrejda, 2013. Unification of regression-based approaches to the analysis of natural selection. Evolution 67:2094–2100.

Schluter, D., 1988. Estimating the form of natural selection on a quantitative trait. Evolution 42:849–861.

Stinchcombe, J. R., A. F. Agrawal, P. A. Hohenlohe, S. J. Arnold, and M. W. Blows, 2008. Estimating nonlinear selection gradients using quadratic regression coefficients: double or nothing? Evolution 62:2435–2440.