Converting regression parameters from linear logistic models using only summary statistics

Introduction

Many traits that we might like to include in the basis are continuous and therefore we only have access to published summary statistics for a linear regression of the form

$$Y = \alpha + \beta X + \epsilon$$

.

We therefore need a method to convert linear regression summary statistics from these studies to their logistic regression counterparts including log(OR) or β coefficients and it's standard error - σ_β. The code below first simulates some continuous and allelic data for a single biallelic SNP with various known parameters. We then fit a linear model and use this to estimate parameters. Next we dichotomise the trait by grouping those individuals whom $Y > \bar{y}$ and then fit a logistic model to estimate β and σ_β.

Next we assume that genotype and trait data is unavailable but that we have access to population allele frequency data for the SNP (MAF), N the sample size, β the linear regression coefficient it standard error σ_β. We then demonstrate that these are suffcient to compute $\bar{x}$ and $\bar{y}$ an estimate of the standard error of ϵ or σ_ϵ. We compare these values to their empirical counterparts to validate the method.

Using a mixed model frame work we then use these to estimate the log(OR) or *beta* coefficient and associated standard error. We validate these metrics against the parameters obtained from our logistic regression fit where we had access to the genotype and trait values.

Method/Results

Simulation setup

```
library(ggplot2)
```

```
## Warning: package 'ggplot2' was built under R version 3.3.2
```

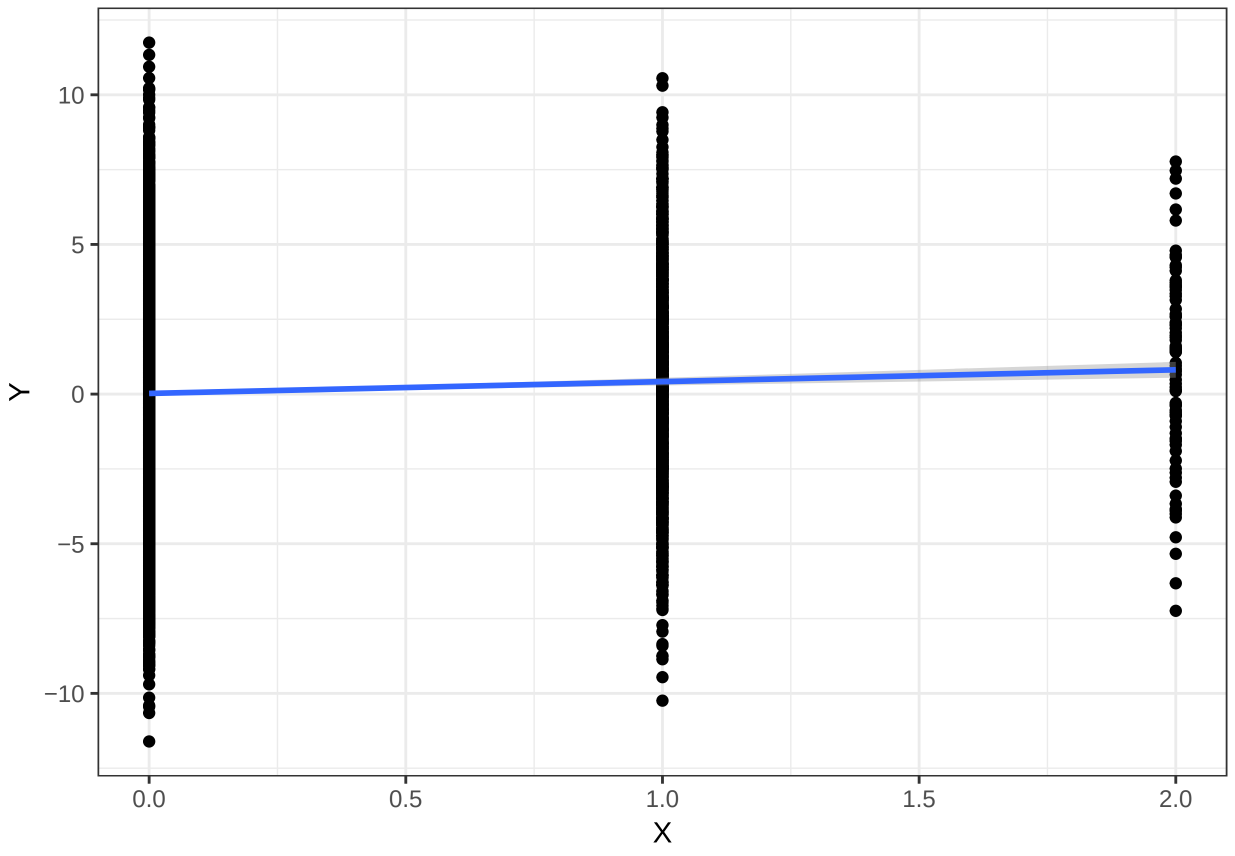
```
set.seed(666)
N<-10000
maf<-0.1
beta<-0.4
p.val<-5e-8
beta.se<-beta/abs(qnorm(p.val/2))
```

Here we simulate a continuous trait with slope $\beta = 0.4$ in a sample size of $N = 10$^{4} in a single biallelic SNP with maf$= 0.1$. This implies a $\sigma_\beta = 0.073$ at the significance level of 5×10^{-8}.

```
#simulate genotypes
X<-rbinom(N,2,maf)
Xbar<-mean(X)
se.e<-beta.se * sqrt(sum((X-Xbar)^2))
#simulate y for a given beta and s
Y=X*beta+rnorm(N,mean=0,sd=se.e)
```

Plot data with linear regression fit

```r
ggplot(data.frame(x=X,y=Y),aes(x=X,y=Y)) + geom_point()  +
    geom_smooth(method='lm',formula=y~x) + theme_bw()
```



Coefficients of the linear model

```r
mod<-lm(Y~X)
summary(mod)
```

```
##
## Call:
## lm(formula = Y ~ X)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -11.6285  -2.0678   0.0346   2.0804  11.7230
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.02165    0.03425   0.632    0.527
## X            0.39480    0.07299   5.409 6.47e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

```
## Residual standard error: 3.088 on 9998 degrees of freedom
## Multiple R-squared:  0.002918,   Adjusted R-squared:  0.002818
## F-statistic: 29.26 on 1 and 9998 DF,  p-value: 6.473e-08
```

Coefficients for the logistic model

```
Yb<-ifelse(Y>mean(Y),1,0)
lmod<-glm(Yb~X,family="binomial")
summary(lmod)
```

```
##
## Call:
## glm(formula = Yb ~ X, family = "binomial")
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.3821  -1.1639   0.9857   1.1909   1.1909
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -0.03183    0.02219  -1.434    0.152
## X            0.25053    0.04766   5.257 1.47e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 13862  on 9999  degrees of freedom
## Residual deviance: 13834  on 9998  degrees of freedom
## AIC: 13838
##
## Number of Fisher Scoring iterations: 3
```

Computing log(OR) using summary stats from linear model

First estimate $\bar{x}$ using just maf, and sample size.

```
raf<-1-maf
## Brute force
wtscore<-function(N,r,w=0) N * r^2 * w
hetscore<-function(N,r,w=1) 2 * N * r * (1-r) * w
homscore<-function(N,r,w=2) N * (1-r)^2 *w
estXbarHat<-(wtscore(N,raf) + hetscore(N,raf) + homscore(N,raf))/N
## if we do the algebra then xbarhat simplifies to 2 * (1-raf) or 2 * maf
XbarHat<-2*maf
```

Actual sample $\bar{x}$ is 0.2, from brute force 0.2 and from simplification 0.2. So far so good.

Next estimate $\bar{y}$, which I think is the value of $Y|X = \bar{x}$ or $\hat{y} = \beta\bar{x}$

```
YbarHat<-beta * XbarHat
```

Actual sample $\bar{y}$ is 0.1 our estimate is 0.08.

For interest want to check that

$$\bar{y} = E(y) = \sum_i E(Y|X_i)P(X_i)$$

i.e

$$\bar{y} = \frac{\beta(N_{\text{het}} + 2N_{\text{hom}})}{N}$$

```
n.wt<-wtscore(N,raf,1)
n.het<-hetscore(N,raf,1)
n.hom<-homscore(N,raf,1)
estYbar<-(beta * (n.het + (2 * n.hom)))/N
```

Using this method the estimated value of $\bar{y}$ is 0.08.

Next we want to estimate the sum of squares for X (SSX) or $\sum_i (X_i - \bar{x})^2$

```
SSX.wt<-sum(n.wt * XbarHat^2)
SSX.het<-sum((rep(1,n.het)-rep(XbarHat,n.het))^2)
SSX.hom<-sum((rep(2,n.hom)-rep(XbarHat,n.hom))^2)
SSXHat=SSX.wt + SSX.het + SSX.hom
```

Actual sample SSX is 1789.91 estimate is 1796.12. Looking good !

An alternative method other than brute force for computing SSX is derived from the fact that

$$SSX = (N-1)\sigma_X^2$$

which simplifies to

$$SSX = 2\hat{m}(N-1)(1-\hat{m})$$

Let's check this result.

```
estSSXHat = 2 * maf * (N-1) * (1-maf)
```

The estimate of SSX this way is 1799.82. Slight difference between brute force, but that could be numerical errors in brute force method.

Next compute $\hat{\sigma_\epsilon}$ that is our estimate of the standard error for ϵ. We compute this using the the given value of σ_β. As we know that $\sigma_\epsilon = \sigma_\beta \times \sqrt{\sum_i (X_i - \bar{x})^2}$. We have estimated SSX above and so have all we need.

```
se.eHat<-beta.se * sqrt(estSSXHat)
```

The actual value of σ_ϵ is 3.1 our estimated value is 3.1. Looking good !

Next we need to estimate the Odds of $(Y > \bar{y}|X = x)$ we can then compute the OR as $\frac{Odds(Y>\bar{y}|X=1)}{Odds(Y>\bar{y}|X=0)}$

```
## next estimate P(Y>ybar | X =x)

calcP<-function(X,beta,SE.epsilon,val=YbarHat){
    pnorm(val,mean=X*beta,sd=SE.epsilon,lower.tail = FALSE)
}

calcOdds<-function(X,beta,SE.epsilon,val=YbarHat){
    #num<-pnorm(val,mean=X*beta,sd=SE.epsilon,lower.tail = FALSE)
    num<-calcP(X,beta,SE.epsilon,val)
    num/(1-num)
}

lod_Y_X1<-log(calcOdds(1,beta,se.eHat))
lod_Y_X0<-log(calcOdds(0,beta,se.eHat))
```

```
lodHat<-lod_Y_X1-lod_Y_X0
## get estimated log(OR) from fit
fitLod<-summary(lmod)[["coefficients"]][,1][2]
```

Actual log(OR) or β from logistic regression fit is 0.25 our estimate is 0.21.

Finally attempt to compute the standard error or σ_β for our $\hat{\beta}$

$$\sigma_\beta = \sqrt{\frac{1}{\text{ploidy}}}\sqrt{\frac{1}{N_0 + N_1}}\sqrt{\frac{1}{a} + \frac{1}{b} + \frac{1}{c} + \frac{1}{d}}$$

```
ca<-function(n0,f){
    n0*(1-f)
}

cb<-function(n0,f){
    n0*f
}

cc<-function(n1,a,b,theta){
    (n1*a)/(a+(b*theta))
}

cd<-function(n1,a,b,theta){
    (n1*b)/(a+(b*theta))
}

gamma_hat_maf<-function(n0,n1,f,theta){
    n<-n0+n1
    a<-ca(n0,f)/n
    b<-cb(n0,f)/n
    c<-cc(n1,a,b,theta)/n
    d<-cd(n1,a,b,theta)/n
    recip.sm<-do.call('cbind',lapply(list(a,b,c,d),function(fi) 1/fi))
    return(sqrt(rowSums(recip.sm)))
}

## compute number of 'cases' i.e. the the number of people with Y>ybar
n.cases<-round(sum(calcP(0,beta,se.eHat) * n.wt,calcP(1,beta,se.eHat) *
                       n.het,calcP(2,beta,se.eHat) *n.hom))
n.controls<-N-n.cases
#this makes sense given that we expect a normal distribution so in
#future can shorcut but N/2 for the special case where we want Y > ybar
## next compute the variance component of beta due to allele freq
ploidy<-2
var_maf<-gamma_hat_maf(n.controls,n.cases,maf,exp(lodHat))
var_ss<-sqrt(1/N)
estSigmaBeta<-sqrt(1/ploidy) * var_maf * var_ss
fitSigmaBeta<-summary(lmod)[['coefficients']][,2][2]
```

From logistic regression fit we estimate $\sigma_\beta = 0.048$ our estimate using just MAF, $\hat{\theta}$ and Sample size is 0.047. Looking good.

Next examine the P value that we get from this compared to the original P

```
estZ<-lodHat/estSigmaBeta
estP<-2*(pnorm(estZ,lower.tail = FALSE))
```

Conclusions

Our input p-val was 5×10^{-8} however we end up with 1.5×10^{-5} so a considerable loss of power as expected when dichotomising a continuous trait. It remains to be seen whether this loss in power is offset by the ability to add/project quantitative traits onto the basis.