Implementation of Interaction Tests

November 25, 2017

Contents

1 Quantitative trait (linear regression) **1**
1.1 Wald test for linear regression 1
1.2 Score test for linear regression 2

2 Dichotomous trait (logistic regression) **3**
2.1 Wald test for logistic regression 3
2.2 Score test for logistic regression 4

1 Quantitative trait (linear regression)

1.1 Wald test for linear regression

```
# 1. generate the data
n=1000
x1 = rnorm(n,1,1)
x2 = rnorm(n,2,3)
y = (1 + 2*x1 +3*x2) + rnorm(n)

mod = glm(y~x1 + x2 + I(x1*x2))
mod.null = glm(y~x1 + x2)

# define wald function -----
wald.linear = function(x1,x2,y){
B_san = matrix(0,nrow=4,ncol=4)
b_fit = matrix(coef(mod),ncol=1)
res = residuals(mod) # res = y - matrix(x%*%b_fit,ncol=1)
Vi = sum(res^2)/(n-4)
x = matrix(cbind(rep(1,n),x1,x2,x1*x2),ncol=4,byrow=F)
Oinv = solve(t(x) %*% x / Vi)
V_mb = Oinv

for (i in 1:n){
  xi = matrix(x[i,],nrow=1)
  B_san = B_san + t(xi)%*%xi * res[i]^2 / Vi^2
}

V_san = Oinv %*% B_san %*% Oinv
Wald_mb = b_fit[4]^2/V_mb[4,4]
Wald_san = b_fit[4]^2/V_san[4,4]
p.wald.mb = 1-pchisq(Wald_mb,df = 1)
p.wald.san = 1-pchisq(Wald_san,df = 1)
p.wald.pack.mb =  coef(summary(mod))[4,4]
p.wald.pack.san = coeftest(mod,df=n-4,vcov. = sandwich)['I(x1 * x2)',4]
return(data.frame(p.wald.mb = p.wald.mb,p.wald.san = p.wald.san,
```

```
            p.wald.pack.mb = p.wald.pack.mb, p.wald.pack.san = p.wald.pack.san# ,
            # V.mb = V_mb[4,4],V.san = V_san[4,4],stat.mb = Wald_mb,stat.san = Wald_san
       ))
}

# compare with package
myresult = wald.linear(x1,x2,y)
kable(myresult,caption = 'Compare Wald test function with package for linear regression')
```

Table 1: Compare Wald test function with package for linear regression

p.wald.mb	p.wald.san	p.wald.pack.mb	p.wald.pack.san
0.0706476	0.0835776	0.0709492	0.0838874

1.2 Score test for linear regression

```
score.linear = function(x1,x2,y){
x0 = matrix(cbind(rep(1,n),x1,x2),ncol=3)
x3 = matrix(x1*x2,ncol=1)
b0_fit = matrix(coef(mod.null),ncol=1)
res0 = y - matrix(x0%*%b0_fit,ncol=1)
Vi = sum(res0^2)/(n-4)
U3 = sum(x3*res0) / Vi

t1 = matrix(0,nrow=1,ncol=3);t2 = matrix(0,nrow=3,ncol=3)
D_mb = matrix(0,ncol=4,nrow=4);D_san = matrix(0,ncol=4,nrow=4)
for(i in 1:n){
  x0i = matrix(x0[i,],nrow=1)
  x3i = x3[i]
  xi = matrix(c(x0i,x3i),nrow=1)
  t1 =  t1 + x3i * x0i / Vi
  t2 = t2 + t(x0i) %*% x0i / Vi
  D_mb = D_mb + t(xi) %*% xi / Vi
  D_san = D_san + t(xi) %*% xi * res0[i]^2 / Vi^2
}
D_mb = D_mb
A = matrix(c(-t1 %*% solve(t2),1),nrow=1)

vs_mb = A %*% D_mb %*% t(A)
Score_mb = U3^2/vs_mb
p.score.mb = 1 - pchisq(Score_mb,df = 1) # model-based score test
p.score.pack.mb = anova(mod,mod.null,test='Rao')[2,'Pr(>Chi)'] # model-based score test by package

vs_san = A %*% D_san %*% t(A)
Score_san = U3^2/vs_san
p.score.san = 1 - pchisq(Score_san,df = 1) # sandwich score test
return(data.frame(p.score.mb = p.score.mb,p.score.san = p.score.san,
            p.score.pack.mb = p.score.pack.mb))
}
```

```
# compare with package
myresult = score.linear(x1,x2,y)
kable(myresult,caption = 'Compare Score test function with package for linear regression')
```

Table 2: Compare Score test function with package for linear regression

p.score.mb	p.score.san	p.score.pack.mb
0.0711095	0.0959338	0.0706476

2 Dichotomous trait (logistic regression)

2.1 Wald test for logistic regression

```
# 1. generate the data
logit.inv = function(x) 1/(1+exp(-x))
n=1000
x1 = rnorm(n,1,1)
x2 = rnorm(n,2,3)
y = apply(matrix(1 + 2*x1 +3*x2,ncol=1),1,function(x) rbinom(1,1,logit.inv(x)))

mod = glm(y~x1 + x2 + I(x1*x2),family = binomial)
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
mod.null = glm(y~x1 + x2,family = binomial)
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
# define wald function -----
wald.logit = function(x1,x2,y){
B_san = matrix(0,nrow=4,ncol=4)
b_fit = matrix(coef(mod),ncol=1)
res = residuals(mod) # res = y - matrix(x%*%b_fit,ncol=1)
x = matrix(cbind(rep(1,n),x1,x2,x1*x2),ncol=4,byrow=F)
mu_hat = matrix(apply(x%*%b_fit,1,logit.inv),ncol=1)
res = y - mu_hat
O = matrix(0,ncol=4,nrow=4)

for (i in 1:n){
  xi = matrix(x[i,],nrow=1)
  O = O + t(xi) %*% xi * mu_hat[i] * (1-mu_hat[i])
  B_san = B_san + t(xi)%*%xi * res[i]^2
}

Oinv = solve(O)
V_mb = Oinv
V_san = Oinv %*% B_san %*% Oinv
Wald_mb = b_fit[4]^2/V_mb[4,4]
Wald_san = b_fit[4]^2/V_san[4,4]
p.wald.mb = 1-pchisq(Wald_mb,df = 1)
p.wald.pack.mb = coef(summary(mod))[4,4]
```

```
p.wald.san = 1-pchisq(Wald_san,df = 1)
p.wald.pack.san = coeftest(mod,df=n-4,vcov. = sandwich)['I(x1 * x2)',4]

return(data.frame(p.wald.mb = p.wald.mb,p.wald.san = p.wald.san,
            p.wald.pack.mb = p.wald.pack.mb, p.wald.pack.san = p.wald.pack.san
       ))
}
# compare with package
myresult = wald.logit(x1,x2,y)
kable(myresult,caption = 'Compare Wald test function with package for logsitic regression')
```

Table 3: Compare Wald test function with package for logsitic regression

p.wald.mb	p.wald.san	p.wald.pack.mb	p.wald.pack.san
0.63447	0.6460479	0.6344695	0.6461478

2.2 Score test for logistic regression

```
# score -------
score.logit = function(x1,x2,y){
x0 = matrix(cbind(rep(1,n),x1,x2),ncol=3)
x3 = matrix(x1*x2,ncol=1)
b0_fit = matrix(coef(mod.null),ncol=1)
mu0_hat = matrix(apply(x0%*%b0_fit,1,logit.inv),ncol=1)
res = y - mu0_hat
U3 = sum(x3*res)

t1 = matrix(0,nrow=1,ncol=3);t2 = matrix(0,nrow=3,ncol=3)
D_mb = matrix(0,ncol=4,nrow=4);D_san = matrix(0,ncol=4,nrow=4)
for (i in 1:n) {
  x0i = matrix(x0[i,],nrow=1)
  x3i = x3[i]
  xi = matrix(c(x0i,x3i),nrow=1)
  t1 = t1 + x3i * mu0_hat[i] * (1-mu0_hat[i]) * x0i
  t2 = t2 + mu0_hat[i] * (1-mu0_hat[i]) * t(x0i) %*% x0i
  D_mb = D_mb + t(xi) %*% xi * mu0_hat[i]*(1-mu0_hat[i])
  D_san = D_san + t(xi) %*% xi * res[i]^2
}

A = matrix(c(-t1 %*% solve(t2),1),nrow=1)
vs_mb = A %*% D_mb %*% t(A)
vs_san = A %*% D_san %*% t(A)

Score_mb = U3^2/vs_mb
p.score.mb = 1 - pchisq(Score_mb,df = 1) # model-based score test
p.score.pack.mb = anova(mod,mod.null,test='Rao')[2,'Pr(>Chi)'] # model-based score test by package

vs_san = A %*% D_san %*% t(A)
Score_san = U3^2/vs_san
p.score.san = 1 - pchisq(Score_san,df = 1) #sandwich score test
```

```
return(data.frame(p.score.mb = p.score.mb,p.score.san = p.score.san,
            p.score.pack.mb = p.score.pack.mb))
}
# compare with package
myresult = score.logit(x1,x2,y)
kable(myresult,caption = 'Compare Score test function with package for logsitic regression')
```

Table 4: Compare Score test function with package for logsitic regression

p.score.mb	p.score.san	p.score.pack.mb
0.6341355	0.6325502	0.6341372