

Generalized linear models and Performance evaluations

GWAS data from genetics lab

Data

```
genos = as.matrix(read.table("./genos.txt"))
phenos = as.matrix(read.table("./phenos.txt"))
g1 = ifelse(genos[,1]==0,0,1)
#table(g1)
g2 = ifelse(genos[,2]==0,0,1)
#table(g2)
#table(g1,g2)
#continuous covariate: phenotype
boxplot(phenos~g1,xlab="genotype",ylab="phenotype")
```

Fit linear model

```
#this is review
mod1 = lm(phenos~g1) #see below for glm()
summary(mod1)
boxplot(phenos~g2,xlab="genotype",ylab="phenotype")
boxplot(phenos~g1*g2)
summary(lm(phenos~g1+g2))
summary(lm(phenos~g1+g2+g1*g2))
#see also: model selection for picking best model
```

Fit linear model using glm and normally distributed error

```
mod1 = glm(phenos~g1,family="gaussian")
summary(mod1)
```

Flip and fit logistic model for binary outcome

```
mod2 = glm(g1~phenos,family="binomial")
summary(mod2)
#predicted probability of g1=1 given phenotype
plot(phenos,g1)
```

```

points(phenos,mod2$fitted.values,pch="p")
#odds ratio
exp(mod2$coef[2])
#adjust for g2
mod3 = glm(g1~phenos+g2,family="binomial")
summary(mod3)
#odds ratio
exp(mod3$coef)
#predicted probability of g1=1 given phenotype and g2
plot(phenos,mod3$fitted.values,pch=g2)
summary(mod3$fitted.values-mod2$fitted.values)

```

Logit function for vector of mu values

```

mu = seq(0,1,length=100)
plot(log(mu/(1-mu)),mu,xlab="Logit(mu)",ylab="mu=E[Y]=Pr(Y=1)",)

```

Performance evaluation

Using the caret package

```

library(caret)
#predict g1=1 if estimated probability > 0.9
g1hat = ifelse(mod2$fitted.values>0.9,1,0)
confusion = table(g1,g1hat)
confusion
confusionMatrix(as.factor(g1),as.factor(g1hat))
#note: treats 0=positive
#see: https://cran.r-project.org/web/packages/caret/vignettes/caret.html
# includes model selection with cross-validation
#manual: https://topepo.github.io/caret/train-models-by-tag.html
# includes many more model types

```

By hand

```

#example: enhancer predictions on chr20
#confusion matrix
tab = c(75,4945,50,7417)
#performance statistics
recall=function(t){t[1]/(t[1]+t[3])}
specificity=function(t){t[4]/(t[2]+t[4])}
fpr=function(t){1-specificity(t)}
precision=function(t){t[1]/(t[1]+t[2])}
fdr=function(t){1-precision(t)}

```

Apply functions to confusion matrix

```
recall(tab)
specificity(tab)
fpr(tab)
precision(tab)
fdr(tab)
```

AUC plots

```
recalls=recall(tab)
fprs=fpr(tab)
precisions=precision(tab)
fdrs = fdr(tab)
```

Increase true positives

```
tab2 = c(80,5440,45,6922)
recalls = c(recalls, recall(tab2))
fprs = c(fprs, fpr(tab2))
precisions = c(precisions, precision(tab2))
fdrs = c(fdrs, fdr(tab2))
```

Max number of true positives

```
tab3 = c(125,12362,0,0)
recalls = c(recalls, recall(tab3))
fprs = c(fprs, fpr(tab3))
precisions = c(precisions, precision(tab3))
fdrs = c(fdrs, fdr(tab3))
```

No true positives

```
tab4 = c(0,0,125,12362)
recalls = c(recalls, recall(tab4))
fprs = c(fprs, fpr(tab4))
precisions = c(precisions, precision(tab4))
fdrs = c(fdrs, fdr(tab4))
```

A few true positives

```
tab5 = c(10,500,115,11862)
recalls = c(recalls, recall(tab5))
fprs = c(fprs, fpr(tab5))
precisions = c(precisions, precision(tab5))
fdrs = c(fdrs, fdr(tab5))
```

One more

```
tab6 = c(25,2400,100,9962)
recalls = c(recalls, recall(tab6))
fprs = c(fprs, fpr(tab6))
precisions = c(precisions, precision(tab6))
fdrs = c(fdrs, fdr(tab6))
```

auROC

```
plot(fprs,recalls,xlim=c(0,1),ylim=c(0,1))
abline(0,1)
```

auPR

```
plot(recalls,precisions,xlim=c(0,1),ylim=c(0,1))
```

auFDR

```
plot(fdrs,recalls,xlim=c(0,1),ylim=c(0,1))
```