Lecture 8 Exercises

Isabel Fulcher

8/20/2018

Install packages

```
library(tidyverse)
library(reshape2)
library(MASS)
```

Bootstrap exercise

We are going to use the bootstrap to get the standard errors from the previous function that you wrote, which likely looked something like,

```
negloglik = function(alpha, X, Y) {
  return(-sum(
    Y*(X%*%alpha)
    -log(1 + exp(X %*% alpha))
  )
  )
}
```

Recall this function corresponds to the following logistic model,

$$\text{logit}(Pr(Y = 1|X_1, X_2)) = \alpha_0 + \alpha_1 X_1 + \alpha_2 X_2$$

Let's create a dataset for this exercise.

```
set.seed(12345)
n <- 100
Sigma <- matrix(c(2,.2,.2,3),2,2)
alpha <- c(.5,.4,-.2)
X <- cbind(1,mvrnorm(n,rep(0,2),Sigma))
Y <- rbinom(n,1,plogis( X%*%alpha))

data <- data.frame(cbind(Y,X[,2:3]))
colnames(data) <- c("Y","X1","X2")
```

1. Perform the bootstrap using $B = 1,000$ replicates for $\hat{\boldsymbol{\alpha}} = (\hat{\alpha}_0, \hat{\alpha}_1, \hat{\alpha}_2)$ and save the estimates from each replicate.

```
B <- 1000

#Option 1: For loop
beta.boot <- matrix(NA,B,3)
for (b in 1:B){

  #take a boostrap sample
  data.boot <- data[sample(n,n,replace=TRUE),]
  X.boot <- as.matrix(cbind(1,data.boot[,2:3]))

  #fit
  fit.optim <- optim(runif(3,0,1),negloglik,Y=data.boot$Y,X=X.boot,method="BFGS")
  beta.boot[b,] <- fit.optim$par

}

# Option 2: sapply function
bootfunc <- function(b,data,n){
  data.boot <- data[sample(n,n,replace=TRUE),]
  X.boot <- as.matrix(cbind(1,data.boot[,2:3]))
  fit.optim <- optim(runif(3,0,1),negloglik,Y=data.boot$Y,X=X.boot,method="BFGS")
  return(fit.optim$par)
}

beta.boot <- t(sapply(1:B,bootfunc,data=data,n=n))

# Option 3: tidyverse purrr?
```

2. Using the results from 1, calculate the bootstrap standard error and 95% confidence intervals.

```
beta.df <- data.frame(beta.boot)
beta.df %>% melt() %>% group_by(variable) %>%
  summarise(sd = sd(value), ci_low = quantile(value,probs=.025),
            ci_up = quantile(value,probs=.975))
```

```
## No id variables; using all as measure variables
```

```
## # A tibble: 3 x 4
##   variable    sd ci_low   ci_up
##   <fct>    <dbl>  <dbl>   <dbl>
## 1 X1       0.228 -0.273 0.609
## 2 X2       0.178  0.218 0.906
## 3 X3       0.117 -0.440 0.00660
```

3. Compare the bootstrap standard error to that from the glm() function.

```
fit.glm <- glm(Y ~ X[,2] + X[,3],data=data,family="binomial")
summary(fit.glm)
```

```
##
## Call:
## glm(formula = Y ~ X[, 2] + X[, 3], family = "binomial", data = data)
##
## Deviance Residuals:
```

```
##     Min       1Q   Median       3Q      Max
## -1.9220  -1.0990   0.6094   1.0289   1.7533
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)   0.1618     0.2203   0.735  0.46257
## X[, 2]        0.5249     0.1694   3.099  0.00194 **
## X[, 3]       -0.2016     0.1128  -1.788  0.07378 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 138.47  on 99  degrees of freedom
## Residual deviance: 124.42  on 97  degrees of freedom
## AIC: 130.42
##
## Number of Fisher Scoring iterations: 4
```

Note that you can also calculate the standard error for for $\hat{\boldsymbol{\alpha}}$ from the optim() function using standard likelihood theory. I have provided the code for this below.

```
fit.optim <- optim(runif(3,0,1),negloglik,Y=data$Y,X=X,method="BFGS",hessian=TRUE)
hess.optim <- fit.optim$hessian
se.optim <- sqrt(diag(solve(hess.optim)))

se.optim
```