# Homework Assignment 5: Bootstrapping Will Continue Until Morale Improves

36-402, Advanced Data Analysis, Spring 2011

SOLUTIONS

1. Answer:

```
library(MASS)
cats.lm1 <- lm(Hwt ~ 0+Bwt,data=cats)  # "0+" sets intercept to zero
summary(cats.lm1)                      # Quick view of results
beta.hat = cats.lm1$coefficients[1]        # Extract beta.hat
se.hat = summary(cats.lm1)$coefficients[2] # Extract se
```

The estimated coefficient is $\hat{\beta} = 3.91$ with standard error of $\hat{se} = 0.044$ (used the summary command). To construct 95% confidence interval for the "true" coefficient $\beta$ under the assumption of Gaussian noise, note that (under this assumption) the sampling distribution of $\hat{\beta}/\hat{se}$ is the $t$-distribution with $n - 1 = 143$ degrees of freedom. The quantiles of the t-distribution in R are given by the `qr` function:

```
t.quantile = qt(0.975, 143) # Determine 97.5th percentile of a t-distribution
CI.lower = round(beta.hat - se.hat*t.quantile, 2)
CI.upper = round(beta.hat + se.hat*t.quantile, 2)
> CI.lower
[1] 3.82
> CI.upper
[1] 3.99
```

Setting the intercept to zero is reasonable because a zero-mass cat should have a zero-mass heart!

2. Answer:

```
plot(density(residuals(cats.lm1)),
  main = "Density estimate for distribution of residuals")
```

See Figure 1. It is easy to see from the plot that the distribution of the residuals is skewed right, unlike the Gaussian. However, since random draws from a Gaussian distribution never look exactly Gaussian due to chance, a formal test of normality might not strongly reject that the data are Gaussian. Indeed, a Shapiro-Wilk test is not especially conclusive:

```
> shapiro.test(residuals(cats.lm1))
        Shapiro-Wilk normality test
data:  residuals(cats.lm1)
W = 0.9819, p-value = 0.05408
```

The p-value is small enough to reject the hypothesis of normality with 90% confidence but not with 95% confidence.

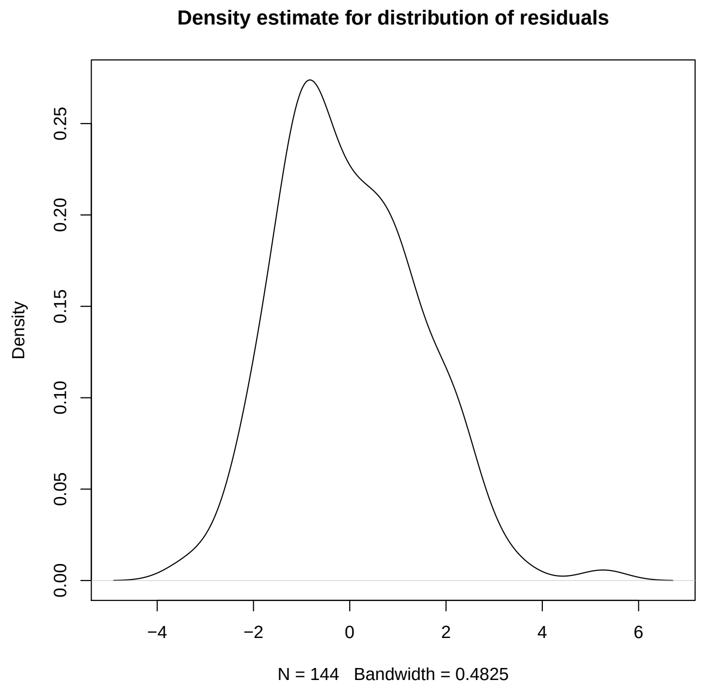

Figure 1:

An alternative test (not shown) would be to fit a Gaussian to the residuals by maximum likelihood, calculate the KS distance, and find $p$-values by simulating from the fitted Gaussian, as was done for the Pareto distribution in the notes to lecture 8.

3. Answer:

```
cats.lm2 <- lm(Hwt~0+Bwt:Sex,data=cats)
```

There should be two regression coefficients since `Sex` is a binary predictor. This gives us two separate slopes, one for male cats and one for female cats.

```
> summary(cats.lm2) # includes the following output:
Coefficients:
         Estimate Std. Error
Bwt:SexF  3.88345    0.08925
Bwt:SexM  3.91461    0.05024
```

To compute confidence intervals we could use the same method as in Problem 1, taking $df = n-2 = 142$ since we are estimating two parameters this time. Or we can take the easy way out and use another R function:

```
> confint(cats.lm2, level = 0.95)
            2.5 %   97.5 %
Bwt:SexF 3.707021 4.059886
Bwt:SexM 3.815296 4.013924
```

An alternative would be to specify the model as

```
cats.lm2b <- lm(Hwt ~ 0+Bwt+Bwt:Sex,data=cats)
```

This would give us one slope which applies to all cats, plus a modification to that slope which applies only to female cats:

```
> summary(cats.lm2b) # includes the following output:
Coefficients: (1 not defined because of singularities)
         Estimate Std. Error
Bwt       3.91461    0.05024
Bwt:SexF -0.03116    0.10242
Bwt:SexM       NA         NA
```

Notice that this says the slope for female cats is $3.91461 - 0.03116 = 3.88345$, so this is the same model as before, just represented slightly differently.

We cannot do

```
lm(Hwt ~ 0 + Bwt*Sex,data=cats)
```

because this will introduce an intercept term, which we don't want.

4. (a) ANSWER: We can use an $F$ statistic to test the hypothesis. First, if we make the assumption that the residuals are Gaussian and i.i.d. for each model, note that the sum of the squared residuals is a chi-squared random variable. For the first model, the degrees of freedom is $n-1 = 143$ and for the second model the degrees of freedom is $n-2 = 142$. Let $SS_1$ denote the sum of squared residuals for the first

model and $SS_2$ denote the sum of squared residuals for the second model. Then the quantity

$$F = \frac{(SS_1 - SS_2)/(143 - 142)}{SS_2/142}$$

is from the $F$ distribution with $(1, 142)$ degrees of freedom. (Look at p. 12 of the textbook by Faraway.) Now if the second (larger) model substantially outperforms the first model, we would expect $SS_2$ to be substantially smaller than $SS_1$. Therefore, a large value of the test statistic F indicates that the second model is better.

Alternately, since we are expanding the model only by one parameter, we could use a $t$-statistic to test the significance of that parameter, given the other. But we can only do this for the second form of the model given in Problem 3, i.e., for `cats.lm2b`.

Alternatively, we could use the log-likelihood ratio test. (See Appendix A of the textbook by Faraway, or pp. 19–20 of the handout for lecture 2.) Suppose we have two models which are "nested", so that one is a special case of the other, and we estimate both by maximum likelihood. Say that the log likelihood of the larger model is $L_{\text{big}}$ and that of the small model is $L_{\text{small}}$, while the number of parameters they have is $d_{\text{big}}$ and $d_{\text{small}}$. Then, under the null hypothesis that the smaller model is true,

$$2(L_{\text{big}} - L_{\text{small}}) \sim \chi^2_{d_{\text{big}} - d_{\text{small}}}$$

when the number of observations $n$ is large. The left hand side is (twice) the log of the likelihood ratio.

If we assume that the fluctuations around the regression have a Gaussian distribution, then the log likelihood is

$$L = -\frac{n}{2}\log 2\pi - n\log\hat{\sigma} - \frac{1}{2}\sum_{i=1}^{n}\frac{(y_i - \hat{y}_i)^2}{\hat{\sigma}^2}$$

but $\hat{\sigma}^2 = n^{-1}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$ so

$$L = -\frac{n}{2}\log 2\pi - n\log\hat{\sigma} - \frac{n}{2}$$

and therefore

$$2(L_{\text{big}} - L_{\text{small}}) = n\log\frac{\hat{\sigma}^2_{\text{small}}}{\hat{\sigma}^2_{\text{big}}}$$

If expanding the model doesn't really help, then the in-sample mean squared error of the larger model, $\hat{\sigma}^2_{\text{big}}$, should be only a little bit better than that of the smaller model, $\hat{\sigma}^2_{\text{small}}$, and this number should have a chi-squared distribution.

(b) Answer: We compute this statistic and it's corresponding p-value in R:

```
### F-test.  Start with defining the sum of squared residuals:
SS1 = sum(cats.lm1$residuals^2)
SS2 = sum(cats.lm2$residuals^2)
F = ((SS1-SS2)/1) / (SS2/142)
## probability of getting a more extreme result under model assumptions:
> 1-pf(F, 1, 142)
[1] 0.7614155
```

Alternatively, we can build a test statistic from the likelihood functions, which is implemented below with less detail:

```
# Actually calculate the LLR by taking residual sums of squares, etc.,
# or short-cut by noticing that SSE = sigma^2 so the only differences are
# n*log(sigma^2 of big model) - n*log(sigma^2 of small model)
# Note: the "sigma" one gets by running summary() on an lm object actually
# does some bias correction, one really wants the un-corrected MLE here.
n = nrow(cats)
twice.llr <- n*(log(mean(residuals(cats.lm1)^2)) - log(mean(residuals(cats.lm2)^2))
> pchisq(twice.llr,df=1,lower.tail=FALSE)
[1] 0.7593824
```

Again, the large p-value suggests that we should not reject the smaller model in favor of the larger one.

(The fact that the $p$-value of the log-likelihood ratio test and that of the partial $F$-test are so close to each other is not an accident. For very large $n$, the two tests become identical.)

5. (a) Answer:

```
resample.cats <- function() {
  sampled.rows <- sample(1:nrow(cats),size=nrow(cats),replace=TRUE)
  new.cats <- cats[sampled.rows,]
}
```

The problem asks us to check this by running `summary` on the output of the function, and seeing that it is close to `summary(cats)`. Let's also make sure that the output is a data frame and that it has the right column names.

```
> is.data.frame(resample.cats())
[1] TRUE
> colnames(resample.cats())
[1] "Sex" "Bwt" "Hwt"
> summary(resample.cats())
 Sex          Bwt              Hwt
 F: 44   Min.   :2.000   Min.   : 6.30
 M:100   1st Qu.:2.300   1st Qu.: 8.80
```

```
         Median :2.700   Median :10.20
         Mean   :2.751   Mean   :10.57
         3rd Qu.:3.100   3rd Qu.:12.05
         Max.   :3.900   Max.   :17.20
> summary(cats)
 Sex         Bwt             Hwt
 F:47   Min.   :2.000   Min.   : 6.30
 M:97   1st Qu.:2.300   1st Qu.: 8.95
        Median :2.700   Median :10.10
        Mean   :2.724   Mean   :10.63
        3rd Qu.:3.025   3rd Qu.:12.12
        Max.   :3.900   Max.   :20.50
```

The summaries are close but not identical, which is appropriate for a stochastic model.

(b) Answer: Just bundle up what we did in Problem 1 into a function.

```
fit.cats.1 <- function(data) {
  fit <- lm(Hwt ~ 0+Bwt, data=data)
  return(coefficients(fit)[1])
}
```

The check:

```
> fit.cats.1(cats)
     Bwt
3.907113  # same as coefficient obtained in part (1)
```

(c) Answer: Follow the template from the lecture notes: apply the estimator to the simulation output many times, and then take the standard deviation.

```
cats.1.se <- function(B) { # B is number of replications
  coefs.on.resamples <- replicate(B,fit.cats.1(resample.cats()))
  return(sd(coefs.on.resamples))
}

> cats.1.se(100)
[1] 0.04213523
> cats.1.se(1000)
[1] 0.04746604  # pretty close to 0.042, which is good; we want consistency here
```

(d) Answer:

```
cats.1.cis <- function(B,alpha) {  # for a 95% CI, set alpha = 0.05
  coef.on.real.data <- fit.cats.1(cats)[[1]]
    # Using [[1]] here instead of [1], or no indexing at all, gets rid of the
    # annoying "Bwt" label.  But the latter is harmless.
  coefs.on.resamples <- replicate(B,fit.cats.1(resample.cats()))
  ci.lower <- 2*coef.on.real.data - quantile(coefs.on.resamples,1-alpha/2)[[1]]
  ci.upper <- 2*coef.on.real.data - quantile(coefs.on.resamples,alpha/2)[[1]]
  return(list(ci.lower=ci.lower,ci.upper=ci.upper))
}
```

Again, this follows the template from the notes: apply the estimator to the simulator many times, take quantiles, and center around the original estimate.

```
# find CI using 100 replicates
> cats.1.ci(100, 0.05)
$ci.lower
[1] 3.823337
$ci.upper
[1] 3.995175

# find CI using 1000 replicates
> cats.1.ci(1000, 0.05)
$ci.lower
[1] 3.819286
$ci.upper
[1] 3.997259
```

(e) Answer: The standard error estimate in problem 1 was 0.044, which is between the two bootstrap estimates we found. The confidence interval bounds are also about the same, to within 0.01. To the extent that we doubt the normality assumption (in problem 2 we found the normality hypothesis to be rejectable with 90% confidence), the bootstrap method may be more trustworthy.

6. (a) Answer: For $k$-fold cross-validation, randomly divide the data into $k$ groups (as near to equal size as arithmetic allows). Pick a group and call it the test data. The remaining groups together are the training data. Estimate the model on the training data. Use the estimated model to predict the response variable for the test data. Compute the mean squared error on the test data. Repeat this process for each group. Averaging the mean squared errors over the $k$ test sets gives an estimate of the model prediction error for new (or “future”) observations. We prefer models with lower prediction error.

(b) Answer:

```
# Function that performs CV and returns estimates of the MSE for each model,
# based on code from HW 2 --- see solutions to that HW
do.nfold.cv = function(nfolds = 5, data){
  case.folds = rep(1:nfolds,length.out=nrow(data))
  case.folds = sample(case.folds)
  fold.mses1 = rep(0,nfolds)
  fold.mses2 = fold.mses1
  for (fold in 1:nfolds) {
    # Split data into training and test data
    train = data[case.folds!=fold,]
    test  = data[case.folds==fold,]

    # Fit both linear models
    cats.lm1 <- lm(Hwt ~ 0+Bwt,data=train)
    cats.lm2 <- lm(Hwt ~ 0+Bwt:Sex,data=train)

    # Predict on the test set
    predictions1  = predict(cats.lm1, newdata=test)
    predictions2  = predict(cats.lm2, newdata=test)

    # Store average squared residuals
    fold.mses1[fold] = mean((test$Hwt - predictions1)^2)
    fold.mses2[fold] = mean((test$Hwt - predictions2)^2)
    }

  # Report the estimated MSE for the first and second models:
  mean(fold.mses1)
  mean(fold.mses2)
  return(c(mean(fold.mses1),  mean(fold.mses2)))
}

# Use the function to estimate the MSE for each model:
> do.nfold.cv(data = cats)
[1] 2.133241 2.167767
```

By CV we estimate that the smaller model has an MSE of 2.13 and the larger model has an MSE of 2.17. Based on this, we choose the smaller model.

(c) Answer: Without knowing the sampling distribution of the estimated MSE (as produced by the function above), we can't say whether the difference in MSE that we observed (2.17 − 2.13) is statistically significant. Luckily we can approximate the sampling distribution using bootstrap replications of the difference.

```
# Here is a function to bootstrap the MSE estimates for both models
bootstrap.MSEs = function(B){
  out = matrix(rep(NA, 2*B), ncol = 2)
  for(ii in 1:B){
    resampled.cats = resample.cats()
    out[ii,] = do.nfold.cv(data = resampled.cats)
  }
  return(out)
}

# Use the function above to do bootstrapping
```

```
B = 100 # number of bootstrap replications
MSE.boots = bootstrap.MSEs(B)
bootstrapped.improvements = MSE.boots[,1] - MSE.boots[,2]
```

Now bootstrapped.improvements is a vector of replications of the "improvement" (as measured by reduction in MSE) of the larger model over the smaller model. We want to test the null hypothesis which is that the two models produce the same MSE. We reject if 0 lies within one of the 2.5% tails of the sampling distribution:

```
> quantile(bootstrapped.improvements, 0.975)
     97.5%
0.05363105 # 0 does not lie in the right tail (not even close)
> quantile(bootstrapped.improvements, 0.025)
       2.5%
-0.05784648 # 0 does not lie in the left tail (not even close)
```

Therefore we cannot reject the hypothesis that the two models do not differ significantly.

Note however that the problem *did not* ask for you to do this bootstrapping.

7. (a) ANSWER: Follow the template in the lecture notes. Leave the independent variables for the regression alone, but re-sample the residuals of the model, add them on to the fitted values of the dependent variable, and call the result the new dependent variable.

```
# A function that simulates new data sets by resampling the residuals
#  from the cats.lm1 model:
resample.residuals.cats.1 <- function() {
  new.cats <- cats
  N = length(residuals(cats.lm1))
  resampling.indices <- sample(1:N,size=N,replace=TRUE)
  new.residuals = residuals(cats.lm1)[resampling.indices]
  new.Hwt <- fitted(cats.lm1) + new.residuals
  new.cats$Hwt <- new.Hwt
  return(new.cats)
}
```

Check that this meets the stated criteria:

```
> is.data.frame(resample.residuals.cats.1())
[1] TRUE
> colnames(resample.residuals.cats.1())
[1] "Sex" "Bwt" "Hwt"
```

We could also check this by looking at the output of `resample.residuals.cats.1`, or even just its first few lines:

```
> head(resample.residuals.cats.1())
  Sex Bwt      Hwt
1   F 2.0 6.692887
2   F 2.0 8.927866
3   F 2.0 5.839330
4   F 2.1 9.437155
5   F 2.1 5.127866
6   F 2.1 6.365021
```

(The `head` function is useful when the output is large but we want to look at the start of it anyway.)

(b) Answer:

```
# This function implements the log-likelihood ratio
#   test described in part (4b)
llr <- function(data) {
  n <- nrow(data)
  cats.1.fit <- lm(Hwt ~ 0+Bwt,data=data)
  cats.1.sigma <- mean((residuals(cats.1.fit))^2)
  cats.2.fit <- lm(Hwt~0+Bwt:Sex,data=data)
  cats.2.sigma <- mean((residuals(cats.2.fit))^2)
  twice.llr <- n*(log(cats.1.sigma) - log(cats.2.sigma))
  return(twice.llr)
}
```

Check that we get the same p-value as in problem 4b:

```
> pchisq(llr(cats),df=1,lower.tail=FALSE)
[1] 0.7593824
```

(c) Answer:

```
# Closely modeled after lecture 8, code example 8
llr.boot.pvalue <- function(B) {
   llr.hat <- llr(cats)
   llr.boot <- replicate(B,llr(resample.residuals.cats.1()))
   p.value <- (1+sum(llr.boot >= llr.hat))/(1+B)
   return(p.value)
}
> llr.boot.pvalue(B = 500)
[1] 0.760479
```

The p-value is large, so we do not reject the null hypothesis that larger model offers no improvement over the smaller model. This result is consistent with our conclusions in problems 4 and 6.

8. Answer: I would recommend that the vet should not include `Sex` in the model. The p-values from problems 4 and 7 both suggest that including `Sex` does not significantly reduce prediction error. Having only considered

the p-values, is entirely possible that including `Sex` in the model changes the predictions (for better or worse), but just at a level that does not show up as statistically significant because we have only 144 cats. The cross-validation done in problem 6 gives a “best guess” as to the effect of including `Sex` on prediction risk; most bootstrap simulations resulted in a higher prediction risk for the model that includes `Sex`. Again, we did not find this increase in risk to be statistically significant, but just in case we are in doubt about whether to include `Sex`, this puts the nail in the coffin: leave `Sex` out of the model.