Regression models

Consider the simple linear regression model

Yj = β0 + β1xj + εj

for j = 1, …, n and εj ~ i.i.d. N(0, σ2). We are going to assume the xj’s are fixed for now.

Here are some of the usual results with this model:

*  and 
* 
*  and 
* 
*  and  have normal distributions
*  and 
*  and 
* E(ej) = 0 and Var(ej) = σ2(1 – hj) where hj is the hat matrix jth diagonal value. Remember that hj is also known as a leverage. In matrix form, **H** = **X**(**X**′**X**)-1**X**′ where **X** = .
* The modified residuals are , and they have a constant variance. The mean of these values is not necessarily equal to 0 so we will end up using  later (these have a mean of 0).
* The standardized residuals are 

What if normality did not hold or what if the constant variance assumption did not hold?

The standard inference procedures may not work.

There are two main ways to take resamples:

1. Model-based resampling
2. Case-based resampling

Model-based resampling

If the model, Yj = β0 + β1xj + εj for j = 1, …, n and εj ~ i.i.d. N(0, σ2), is correct, we can take advantage of the εj coming from the same distribution! This follows a very similar algorithm to what was used as “Approach #3” when testing H0: μ1 – μ2 = 0 vs. Ha: μ1 – μ2 ≠ 0. As you will see, it may be better to call this “semiparametric” instead of nonparametric.

We could resample the ej’s because they are estimates of the εj’s. Instead, it is better to resample from the rj because their variances agree with εj. Remember that Var(ej) = σ2(1 – hj) and Var(rj) = σ2. We will further modify this to resample  because they have a mean of 0 like εj ( does not necessarily equal 0). Note that  so that it is essentially equal to Var(rj) (see p. 6.10 of my STAT 950 notes for a proof)

We keep the  part the same for each resample. Thus,



where  is resampled from the , …, . One resample is . Each resample has the same “design” because the xj are the same as for the original sample.

The expected extensions to this resampling approach are made for multiple linear regression models.

Example: Mammals (mammals.R, mammals.txt)

The purpose of this problem is to use body weight to predict brain weight in n = 62 mammals.

> library(boot)

> #For each mammal, the first number is body weight in

kilograms and the second number is brain weight in

grams.

> mammals<-read.table(file = "C:\\chris\\mammals.txt",

header = TRUE)

> head(mammals)

animal body brain

1 African\_elephant 6654.000 5712.0

2 African\_giant\_pouched\_rat 1.000 6.6

3 Arctic\_Fox 3.385 44.5

4 Arctic\_ground\_squirrel 0.920 5.7

5 Asian\_elephant 2547.000 4603.0

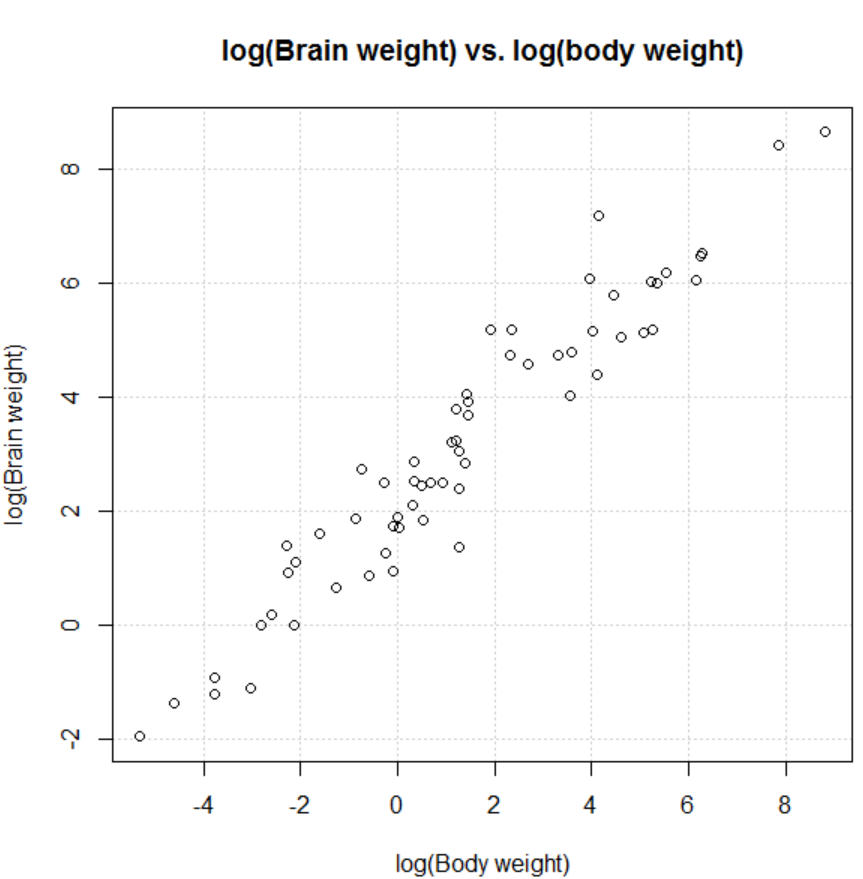
6 Baboon 10.550 179.5

> plot(x = log(mammals$body), y = log(mammals$brain),

main = "log(Brain weight) vs. log(body weight)",

xlab = "log(Body weight)", ylab = "log(Brain

weight)", panel.first = grid())



> mod.fit <- lm(log(brain) ~ log(body), data = mammals)

> summary(mod.fit)

Call:

lm(formula = log(brain) ~ log(body), data = mammals)

Residuals:

Min 1Q Median 3Q Max

-1.71550 -0.49228 -0.06162 0.43597 1.94829

Coefficients:

Estimate Std. Error t value Pr(>|t|)

(Intercept) 2.13479 0.09604 22.23 <2e-16 \*\*\*

log(body) 0.75169 0.02846 26.41 <2e-16 \*\*\*

---

Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.6943 on 60 degrees of freedom

Multiple R-Squared: 0.9208, Adjusted R-squared: 0.9195

F-statistic: 697.4 on 1 and 60 DF, p-value: < 2.2e-16

> names(mod.fit)

[1] "coefficients" "residuals" "effects" "rank"

[5] "fitted.values" "assign" "qr"

"df.residual"

[9] "xlevels" "call" "terms"

"model"

> anova(mod.fit)

Analysis of Variance Table

Response: log(brain)

Df Sum Sq Mean Sq F value Pr(>F)

log(body) 1 336.19 336.19 697.42 < 2.2e-16 \*\*\*

Residuals 60 28.92 0.48

---

Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

> n <- nrow(mammals)

> #Get h\_j

> influence.stat <- lm.influence(mod.fit)

> h.j <- influence.stat$hat

> #Modified residuals

> r.j <- mod.fit$residuals/sqrt(1-h.j)

> mean(r.j)

[1] -0.000895757

> #Get sqrt(MSE)

> sum.fit <- summary(mod.fit)

> names(sum.fit)

[1] "call" "terms" "residuals" "coefficients"

[5] "aliased" "sigma" "df" "r.squared"

[9] "adj.r.squared" "fstatistic" "cov.unscaled"

> sum.fit$sigma #sqrt(MSE)

[1] 0.6942947

The estimated regression model is:

 = 2.13479 + 0.75169log(x)

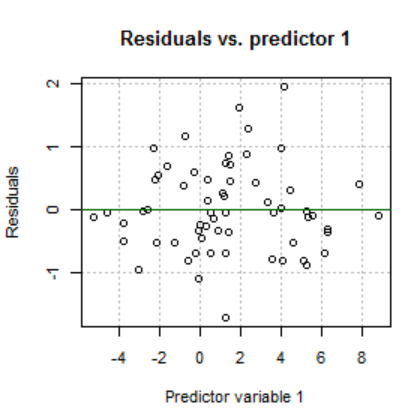
where x = body weight and y = brain weight. Notice  =   
-0.000896 ≠ 0.

I used a function from my STAT 870 class to examine model diagnostics. Below is one of the plots.

> source(file = "C:\\chris\\examine.mod.multiple.final.R")

> examine.mod.multiple.final(mod.fit.obj = mod.fit,

first.order = 1)



There are some “possible” non-constant variance issues.

Finally, let’s use the bootstrap with R = 4999 resamples and the model-based resampling method.

> calc.t.modbased <- function(data, i, mu.hat, x) {

epsilon <- data[i]

y <- mu.hat + epsilon

mod.fit.modbased <- lm(formula = y ~ x)

sum.fit.modbased <- summary(mod.fit.modbased)

c(as.numeric(mod.fit.modbased$coefficients),

sum.fit.modbased$sigma)

#as.numeric() is just used to remove some not

needed labels

}

> #Try it

> calc.t.modbased(data = r.j- mean(r.j), i = 1:n, mu.hat=

mod.fit$fitted.values, x = log(mammals$body))

[1] 2.1348397 0.7516478 0.7035203

> set.seed(8719)

> boot.res.modbased <- boot(data = r.j - mean(r.j),

statistic = calc.t.modbased, R = 4999, sim =

"ordinary", mu.hat = mod.fit$fitted.values, x =

log(mammals$body))

> boot.res.modbased

ORDINARY NONPARAMETRIC BOOTSTRAP

Call:

boot(data = r.j - mean(r.j), statistic = calc.t.modbased, R

= 4999, sim = "ordinary", mu.hat =mod.fit$fitted.values,

x = log(mammals$body))

Bootstrap Statistics :

original bias std. error

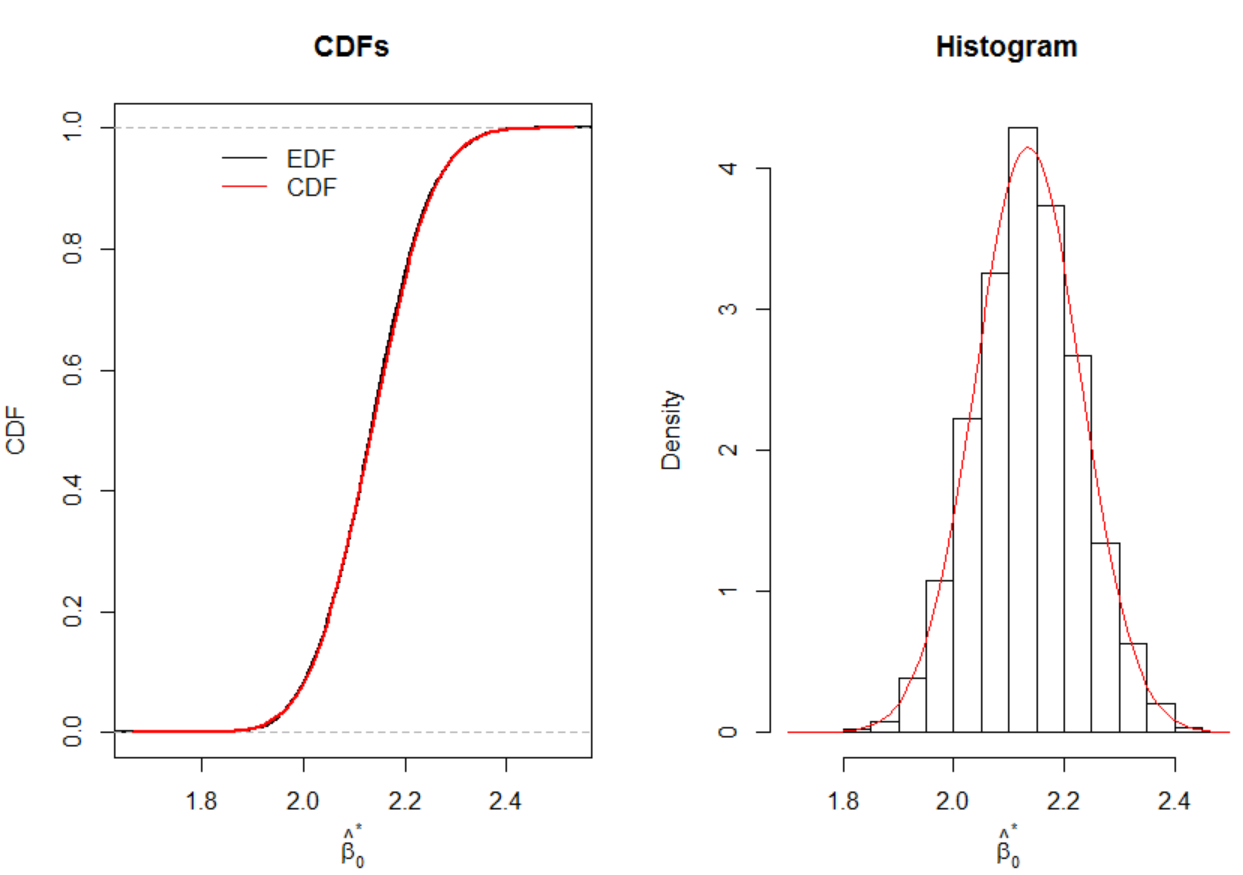
t1\* 2.1348397 -0.0001045672 0.09521836

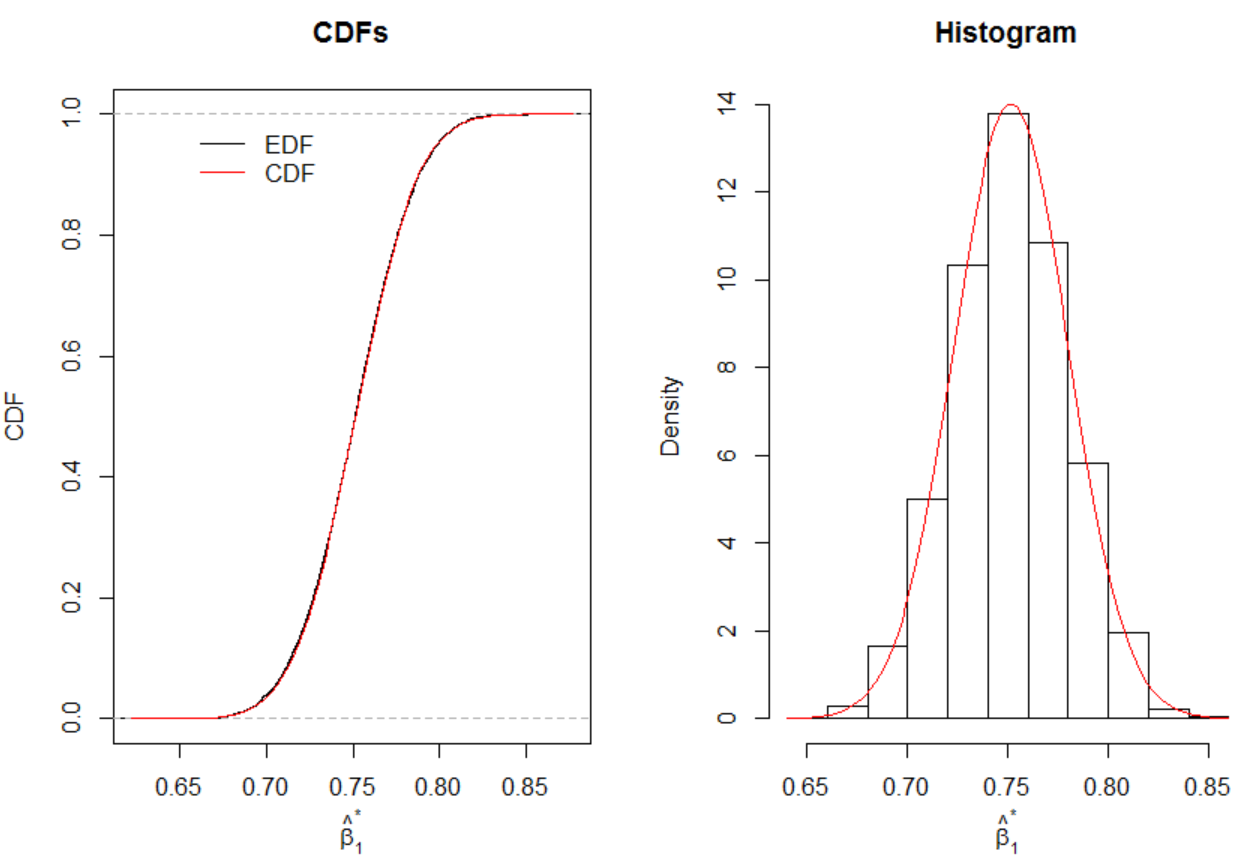
t2\* 0.7516478 -0.0004101954 0.02867706

t3\* 0.7035203 -0.0152763174 0.06720262

Question: Why is the “original” value listed in the t2 row different from the  = 0.75169 calculated earlier?

Below are plots comparing the distribution obtained from the resamples and the estimated normal distribution (see program for code).





Additional calculations for 

> SSx <- var(log(mammals$body))\*(n-1) #SUM((x\_i –x\_bar)^2)

> z.star.beta1 <- (boot.res.modbased$t[,2] -

mod.fit$coefficients[2]) /

sqrt(boot.res.modbased$t[,3]^2/SSx)

#Note: Var(beta\_hat1\*) = s^2\* / SSx

> #Estimated quantiles for z\*

> quantile(x = z.star.beta1, probs = c(0.05, 0.95), type

= 1)

5% 95%

-1.707681 1.670025

> qnorm(p = c(0.05, 0.95)) #Expected quantiles for

epsilon~N(0, sigma^2)

[1] -1.644854 1.644854

The quantiles of the standard normal are not too far away from what resampling gives us.

You should think about what would could be done next:

* C.I. for β1
* C.I. for E(Y)
* C.I. for x (inverse prediction)

Case-based resampling

This type of resampling takes (X,Y) as sampled pairs from a bivariate distribution. Thus, resample pairs from (x1, y1), …, (xn, yn) with replacement.

Discussion of the differences between resampling methods:

* Resampling cases makes no assumption about constant variance. Of course, if the constant variance assumption is correct, case-based resampling will be less efficient than model-based resampling.
* Resampling cases causes different “designs” because the x’s are being resampled. We have information on the specific x’s in our sample. If an x is not included in a resample, we are not taking into account this information. For example, remember one of the first things you learned in a regression class is to not extrapolate beyond the range of your x’s; notice what could happen with case-based resampling

Davison and Hinkley (1997) say “the variation in  will cause some variation in information, but fortunately this is often unimportant in moderately large data sets.”

Example: Mammals (mammals.R, mammals.txt)

> calc.t.cases<-function(data, i) {

d<-data[i,]

mod.fit.cases<-lm(formula = log(brain) ~

log(body), data = d)

sum.fit.cases<-summary(mod.fit.cases)

c(as.numeric(mod.fit.cases$coefficients),

sum.fit.cases$sigma)

}

> #Try it

> calc.t.cases(data = mammals, i = 1:n)

[1] 2.1347887 0.7516859 0.6942947

> set.seed(4121)

> boot.res.cases<-boot(data = mammals, statistic =

calc.t.cases, R = 999, sim = "ordinary")

> boot.res.cases

ORDINARY NONPARAMETRIC BOOTSTRAP

Call:

boot(data = mammals, statistic = calc.t.cases, R = 4999, sim = "ordinary")

Bootstrap Statistics :

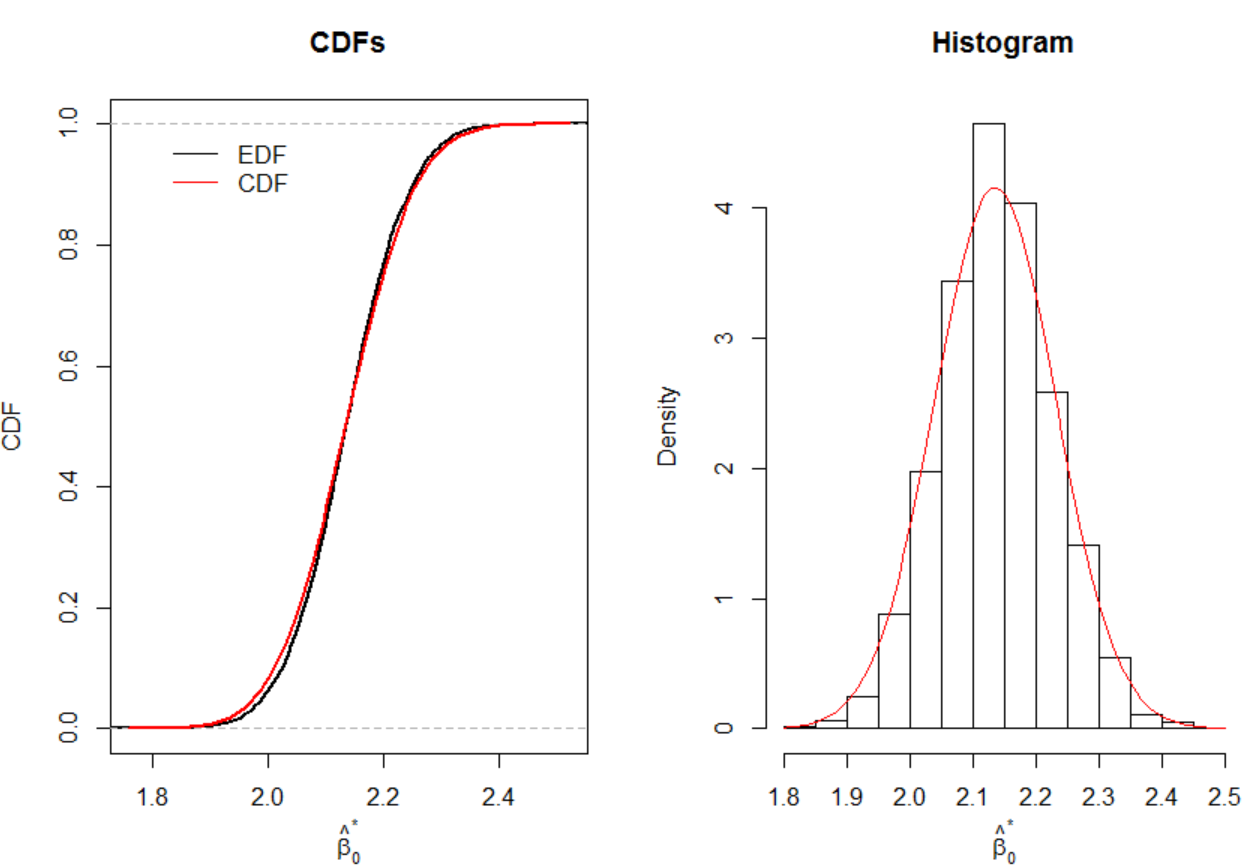
original bias std. error

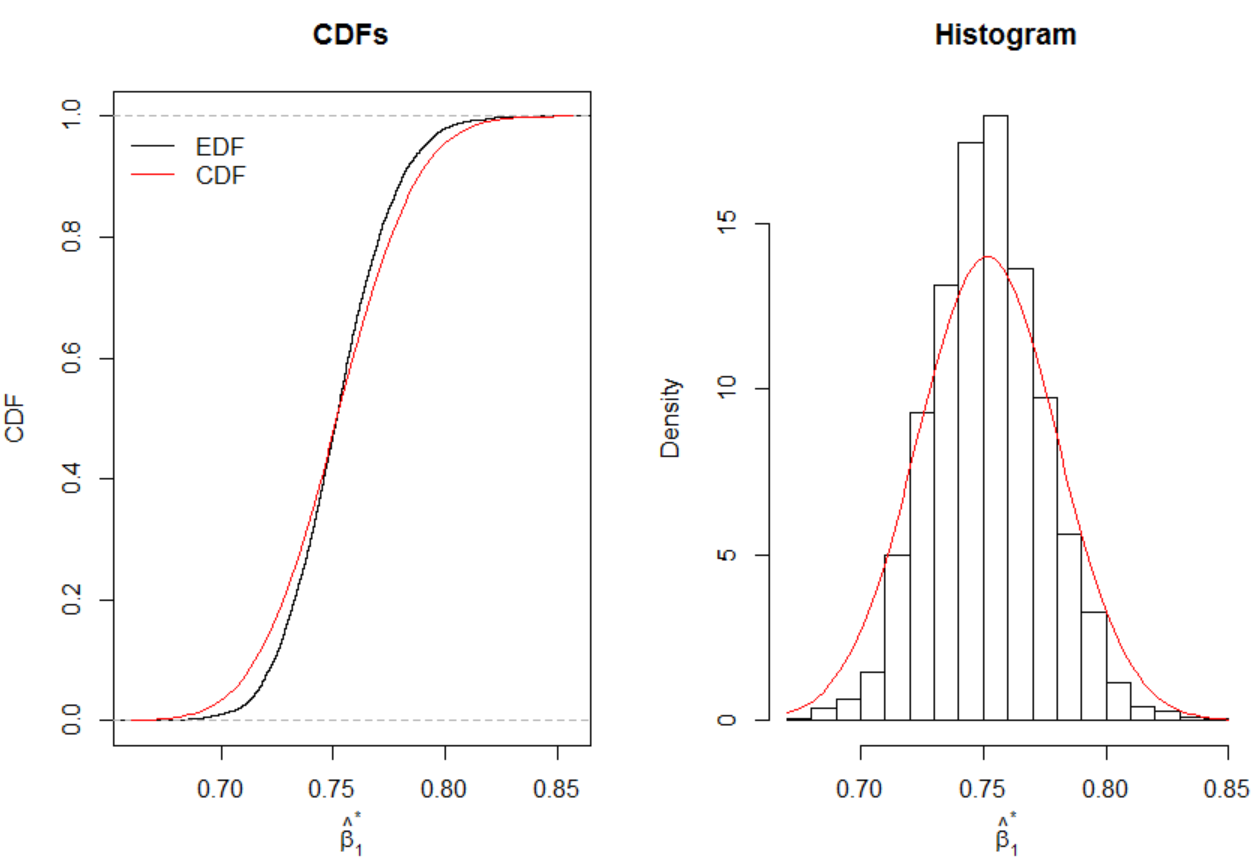
t1\* 2.1347887 0.002459914 0.08866105

t2\* 0.7516859 0.000221622 0.02278218

t3\* 0.6942947 -0.013145460 0.06570832

Below is an examination of the distributions for the regression parameter estimates:





The distributions look close to normal except for possibly a little deviation in the tails for the  plot.

You should think about what would could be done next:

* C.I. for β1
* C.I. for E(Y)
* C.I. for x (inverse prediction)

Prediction intervals

Confidence intervals for E(Y), the mean response at a particular set of explanatory variable values, can be found as one would expect. Prediction intervals for an unknown individual response, say Y+ with explanatory variable vector **x**+, call for some changes to incorporate Y+ being a random variable, not an expected value. The + subscript here is not used to mean something is being summed over. Rather, it is just one way to notational show that this is for some set of explanatory variable values that are not necessarily in the data set.

We can predict  with the point predictor  (=, say). The accuracy of the point predictor is



The distribution of δ can be estimated using the bootstrap! Notice how this is like the distribution of T − θ, denoted by G; we estimate G with the distribution of T\* − t. In this case here,



where  and  come from model-based resampling.

Comments:

*  is estimated using model-based resampling of n mean adjusted modified residuals. Thus, use , where  are resampled from , to obtain the resampled data on which  is found through fitting the regression model.
*  is obtained by a SEPARATE resample from ! Thus, another resample, but of size 1.

Prediction interval (P.I.) for Y+

The usual (1 – 2α)100% P.I. that you have learned about in a regression class has limits of



where p + 1 is the number of β parameters,  is the observed value for , . Stated a different way, the P.I. limits are

 – a1-α and  – aα

where aα is the αth quantile from the distribution of δ. Because the distribution of δ is unknown, the bootstrap is used to approximate it. The basic bootstrap interval limits are

 –  and  – 

Make sure you are understand why there is no “2t” in this interval like we have seen before. The studentized bootstrap interval limits are

 –  and  – 

where . Other intervals could be calculated as well.

Hypothesis testing for regression parameters

Resamples need to be taken under the null hypothesis! Consider the case of a simple linear regression model and performing the test H0: β1 = 0 vs. Ha: β1 ≠ 0.

1. Case-based resampling

Resamples are taken with replacement from the x’s and y’s (independently). This resampling is similar to what we saw with a contingency table and testing for independence. Why is this “case-based” resampling?

For each resample, the FULL model including the resampled xj, needs to be estimated in order to find  for r = 1, …, R. This will provide us a MC simulation estimate of the distribution for . The p-value can then be calculated the usual way.

1. Model-based resampling

The model under H0 is Yj = β0 + 0xj + εj = β0 + εj for j = 1, …, n and εj ~ i.i.d. N(0, σ2). This model results in an estimated  (estimated E(Yj) under H0) of simply  for all j = 1, …, n.

We can find  where  are resampled the usual way. The FULL model including xj, is estimated for each resample in order to find  for r = 1, …, R. This will provide us a MC simulation estimate of the distribution for . The p-value can then be calculated the usual way.

The test statistic would be  or .

For a multiple linear regression model and tests involving one or more regression parameters, the process can be a little more complicated. For example, consider the model

**Y** = **Xβ** + **ε** = **X**0**α** + **X**1**γ** + **ε**

where **β** is (p+1)×1 (the extra 1 is for the intercept parameter), **α** is (q+1)×1, and **γ** is (p-q)×1. This results in writing a

FULL model of **Y** = **Xβ** + **ε**

and a

REDUCED model of **Y** = **X**0**α** + **ε0**.

We then test H0: **γ** = 0 vs. Ha: **γ** ≠ 0.

Under model-based resampling, we can use



where  are the estimated values under the H0 model (reduced model) and  are resampled from  (j = 1, …, n). The  are obtained from fitting the REDUCED model to the observed data. The FULL model is fit to the resampled data with **X** the same as originally obtained in the sample.

With case-based resampling, the resampling under the null hypothesis is a little tricky. In the simple linear regression case, the response and explanatory variables were simply resampled independently with replacement. The same idea can be used here, but the effects of the variables in **X**0 need to be removed from **Y** and **X**1 first.

To remove the linear effect of **X**0 from **Y**, simply perform a regression with **Y** as the response and **X**0 as the explanatory variables. The residuals, **e**0, represent what is left over after the effect of **X**0 has been removed.

To remove the linear effect of **X**0 from **X**1, simply perform a regression with **X**1 as the response and **X**0 as the explanatory variables. The residuals, **X**1.0, say, represent what is left over after the effect of **X**0 has been removed.

Note that this idea is similar to what is used with partial regression plots (see Chapter 10 of my STAT 870 course notes).

When p – q > 1, **X**1.0 will be a n×(p-q) matrix.

There are (p − q) different “response” variables so “multiple” regression (as defined in Chapter 12 of Johnson (1998)) is being performed. Simply, (p-q) different regression models can be fit (one for each “response” variable) to obtain predicted values. Because the residuals are of interest here, one can simply use .

The test statistic calculated is  or  for p – q = 1. When p – q > 1, the usual F-test statistic could be used.

**Final comments about the bootstrap**

1. The parametric bootstrap assumes that F is some named distribution. The estimate of F, , then simply replaces the parameters in the distribution with their observed estimates. For example, the AC data could use F as being Exponential(μ). The distribution for  would be Exponential().   
     
   Resampling is performed from  using the random generation tools available for named distributions. For example, the AC data could use the rexp() function. All of the calculations that we have seen earlier proceed in the same manner using these resamples. Alternatively, exact expressions can be derived for quantities of interest. For example, the AC data would use  ~ Gamma(n, μ/n) and  ~ Gamma(n, /n). This leads to the percentile interval being the α and 1 – α quantiles from a Gamma(n, /n), so there is no need for MC simulation.
2. Cautions about using the bootstrap:
   1. A main reason for the bootstrap not working well is when small changes in F cause large changes in G. The reason can be found through examining statistical functions and the plug-in principle. For example, suppose you want to estimate the distribution of . When a slightly altered sample causes large changes in the statistic, this will greatly affect how well the bootstrap works. A situation where this can occur is with respect to extreme order statistics. Davison and Hinkley (1997) provide a nice example on p. 39 involving Y(n) and my Chapter 2 lecture notes expand upon it. In order for the bootstrap to work, we need smooth statistical functions – small changes in F cause small changes in t(F).
   2. Dependent observations can cause problems. Notice that we resample from  assuming independence. However, there are ways to take into account dependence – see Chapter 8 of Davison and Hinkley (1997).
3. Much of the underlying mathematical development of the bootstrap relies on the use of Edgeworth expansions. One can think of Edgeworth expansions as like a Taylor series expansion for a statistic. However, instead of statistics, Edgeworth expansions are used to approximate probability distributions in terms of a standard normal. These Edgeworth expansions are used to prove the asymptotic accuracy of the distribution based on resamples. Page 517 of Casella and Berger (2002) gives a short introduction to these expansions. Hall’s (1992) Chapter 2 provides a larger introduction to the expansions and then he uses them to prove why the bootstrap works as n→∞.