More on residuals: Non-normality, Non-constant variance, WLS, and Bootstrapping

In this handout we examine more closely the assumption that the errors $\underline{\epsilon}$ in the standard regression model $\underline{y} = X\underline{\beta} + \underline{\epsilon}$ are iid $N(0, \sigma_\epsilon^2)$, and look at some thing to do in case they are not.

1 Non-normal residuals

We have worked with all the residuals

$$E_i = y_i - \hat{y}_i \qquad E_i' = \frac{E_i}{S_E\sqrt{1-h_{ii}}} \qquad E_i^* = \frac{E_i}{S_{E(-i)}\sqrt{1-h_{ii}}}$$

when the normal-errors linear regression model holds, where $S_E^2 = RSS/(n-k-1)$ is the squared residual standard error for the full-data regression, and $S_{E(-i)}^2$ is from the same regression without observation i. When the assumptions of the normal-errors linear regression model hold, the studentized residuals E_i^* has a t_{n-k-2} distribution; it's less clear what the distribution of the standardized residuals is, though we know asymptotically they will be $N(0,1)$. However, each set of residuals is dependent, since in particuar

$$\text{Var}\,(\underline{E}) = \sigma_\epsilon^2(I - H)$$

and the off-diagonal elements of the hat matrix $H = X(X^TX)^{-1}X^T$ are seldom zero. Still, each set of residuals provides a useful tool for assessing the error distribution of the regression.

The Fuel Data

```
> mod3 _ lm(Fuel ~ Dlic + Tax, data=fuel48)
> resids _ residuals(mod3)
> zres _ stdres(mod3)
> tres _ studres(mod3)

> par(mfrow=c(2,3))

> qqenv(resids)
> qqenv(zres)
> qqenv(tres)

> plot(density(resids,width=sqrt(var(resids))),type="l")
> points(resids,rep(0,length(resids)))
> plot(density(zres,width=sqrt(var(zres))),type="l")
> points(zres,rep(0,length(zres)))
> plot(density(tres,width=sqrt(var(tres))),type="l")
> points(tres,rep(0,length(tres)))
```

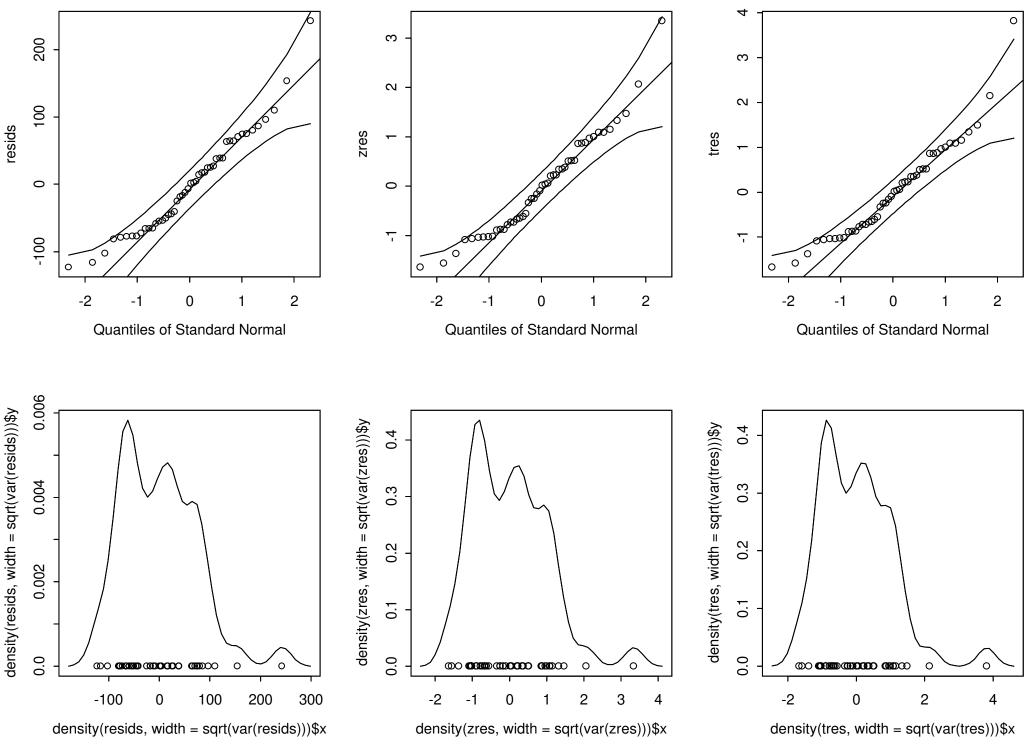

Figure 1: Examining the distribution of the residuals the model `lm(Fuel ~ Dlic + Tax, data=fuel48)`.

The result is in Figure 1. We see again that there isn't much difference between standardized (E_i') and studentized (E_i^*) residuals. More interesting is that there doesn't appear to be much difference between the distributions of the raw residuals E_i and the studentized and standardized residuals.

The QQ plots suggest long right tails and short left tails, and this is confirmed by the density estimates. The studentized residuals seem best at picking out the high outlier around $E^* = 4$ in the QQ plots, but this outlier can be seen in all three density plots.

Another interesting aspect of the density plots is that they suggest two modes in the middle of the residual distribution. This can indicate that there are two different "kinds" of data mixed together in the problem, and one can explore this in the regression by constructing a dummy variable to code for the two nodes separately. Another way to explore this is to look at the identities of the data in the "left mode" and in the "right mode". The valley between the two nodes looks like it occurs at about $E_i' = -0.3$ or so, so let's see if there's anything to this idea:

A quick split of the state names on `zres < (-0.3)` suggests that most of the low-residuals belong to states in the northeast, northwest or midwest:

```
> split(zres,zres < (-0.3)) # without the extra ()'s Splus will think "<-" !
$FALSE:
        ME         NH        VT        IN      IA         MO       ND       SD
 0.8656834 -0.1651178 0.2022272 0.8592405 0.22937 0.03613089 2.065885 1.083199
```

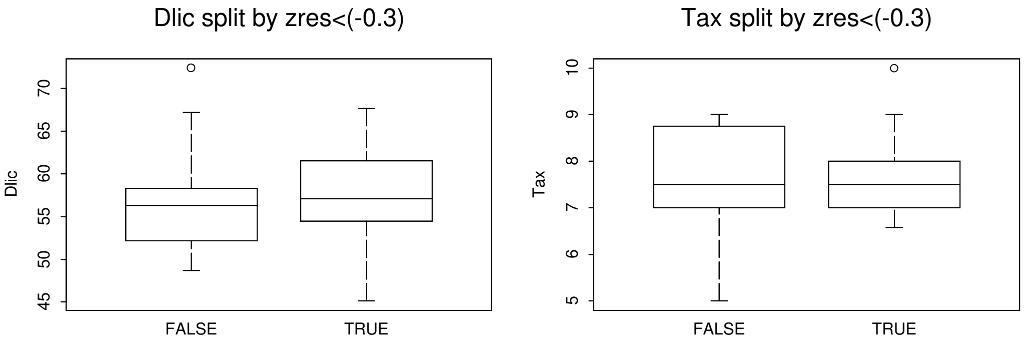

Figure 2: Comparing the regressor values in two tentative "humps" of the residual distribution from the model `lm(Fuel ~ Dlic + Tax, data=fuel48)`.

```
        MD         VA         NC         SC         GA         FL        KY       TN
 0.05726105 1.085256 0.8804769 0.5164972 0.5044255 0.2265553 1.328408 0.51919
       AL         MS          AR         LA          TX        MT          ID        WY
 0.3745097 0.01692816 0.9989273 0.3424751 -0.2497392 1.150291 -0.2533677 3.345335
      NM         AZ         UT         NV
 1.469591 -0.09478661 0.9654467 0.3442131

$TRUE:
       MA        RI          CN        NY         NJ          PA         OH
 -1.557909 -1.639922 -0.6569055 -1.01975 -1.029837 -0.6732776 -1.034664
        IL         MI         WI         MN          NE          KS          DE
 -0.7293698 -1.038478 -0.7848629 -1.061832 -0.619824 -0.8932164 -0.8787991
        WV          OK         CO         WN          OR         CA
 -0.8822003 -0.5571016 -1.087707 -0.3384047 -0.7271349 -1.369626
```

Examining the X variables (see Figure 2) suggests that the low-residual states may have both a somewhat greater proportion of licenced drivers per capita and a possibly higher distribution of fuel taxes—an unusual combination. At any rate this seems interesting enough to explore further.

```
> boxplot(split(fuel48$Dlic,zres < (-0.3)),style.bxp="old")
> title("Dlic split by zres<(-0.3)",ylab="Dlic")
> boxplot(split(fuel48$Tax,zres < (-0.3)),style.bxp="old")
> title("Tax split by zres<(-0.3)",ylab="Tax")
```

2 Non-constant error variance

2.1 A Nice Example

As an example we will consider the data of Ornstein (1976), "The boards and executives of the largest Canadian corporations." *Canadian Journal of Sociology, 1,* 411–437. The variables are:

```
Firm:   Firm number, ranked by assets

Assets: Assets in millions of dollars

Sector: Industrial Sector
    AGR = agriculture, food, light industry
    MIN = mining, metals, etc.
    WOD = wood and paper
    CON = construction
    MAN = heavy manufacturing
    TRN = transport
    MER = merchandizing
    BNK = banks
    FIN = other financials
    HLD = holding companies

Nation: Nation of Control
    CAN = Canada
    US  = United States
    UK  = Britain
    OTH = other foreign

Interlocks:  Number of interlocking director and executive positions
```

where, I suppose, an interlocking director/exectutive position is, for example, a position held on the board of one company by the employee of another company. Boxplots (Figure 3) readily reveal that there appears to be a pattern in the number of interlocks by nation of control of the corporation, and the question is whether the other variables can account for this, or whether it is purely a national effect.

```
> ornstein _ read.table("ornstein.dat")
> dim(ornstein)
[1] 248   5
> names(ornstein) _ scan(what="")
1: Firm
2: Assets
3: Sector
4: Nation
5: Interlocks
6:

> attach(ornstein)
> boxplot(split(Interlocks,Nation),style.bxp="old")
```

Preliminary analysis (Figure 4) suggests some skewing and high-outlier features of the `Assets` variable (which could be anticipated, since it is a monetary amount). We should get rid of this, not because the X variables need to be normally distributed, but because the high `Asset` outliers

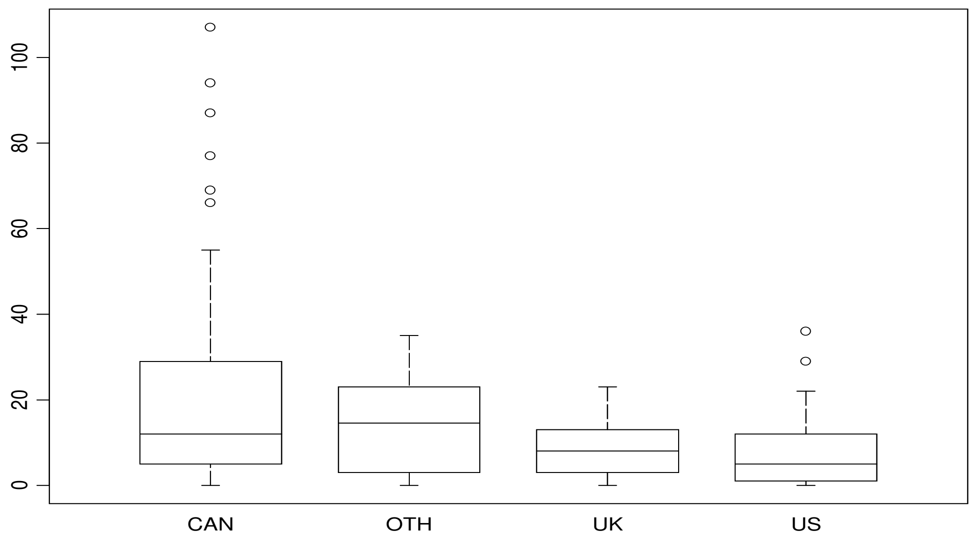

Figure 3: Comparing number of interlocking board/executive positions in firms based in various nations.

will have high leverage on the regression. One might try a square-root transformation but Figure 4 shows that a log transformation works even better.

```
> par(mfrow=c(2,2))
> plot(density(Assets,width=sqrt(var(Assets))),type="l")
> points(Assets,rep(0,248))
> plot(density(sqrt(Assets),width=sqrt(var(sqrt(Assets)))),type="l")
> points(sqrt(Assets),rep(0,248))
> plot(density(log(Assets),width=sqrt(var(log(Assets)))),type="l")
> points(log(Assets),rep(0,248))
```

2.2 An Initial Analysis

The most straightforward regression to consider here is to toss all the variables in and see what happens. The diagnostic plots are like the standard `plot(myfit)` diagnostics, but I have substituted in a plot of studentized residuals and a `qqenv` plot so you can see what these look like. The visual effect is the same if I use all raw residuals.

```
> myfit _ lm(Interlocks ~ log(Assets) + Sector + Nation)
> par(mfrow=c(2,3))
> plot(myfit,residuals=studres(myfit))
> plot(fitted(myfit),studres(myfit))
> abline(h=0)
```

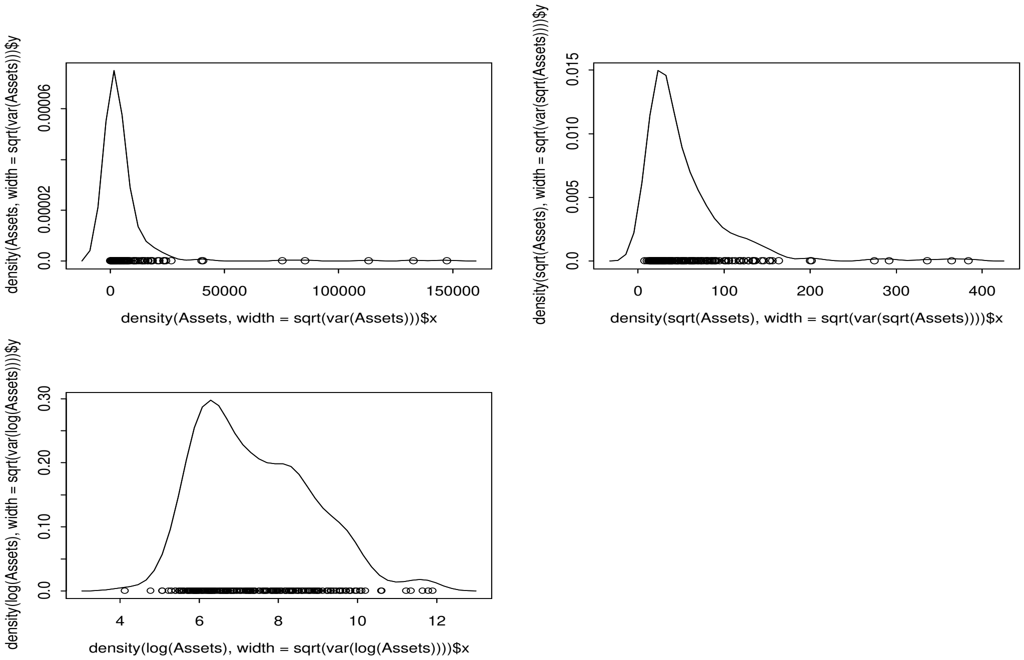

Figure 4: Fixing skewing and high outliers in the `Assets` variable of the interlocks data, in the hopes of reducing the number of data points with high leverage.

```
> plot(myfit,ask=T)

Make a plot selection (or 0 to exit):

1: plot: All
2: plot: Residuals vs Fitted Values
3: plot: Sqrt of abs(Residuals) vs Fitted Values
4: plot: Response vs Fitted Values
5: plot: Normal QQplot of Residuals
6: plot: r-f spread plot
7: plot: Cook's Distances
Selection: 3

[etc.]

Selection: 4

[etc.]

Selection: 0

> qqenv(studres(myfit))

> plot(myfit,ask=T)
```

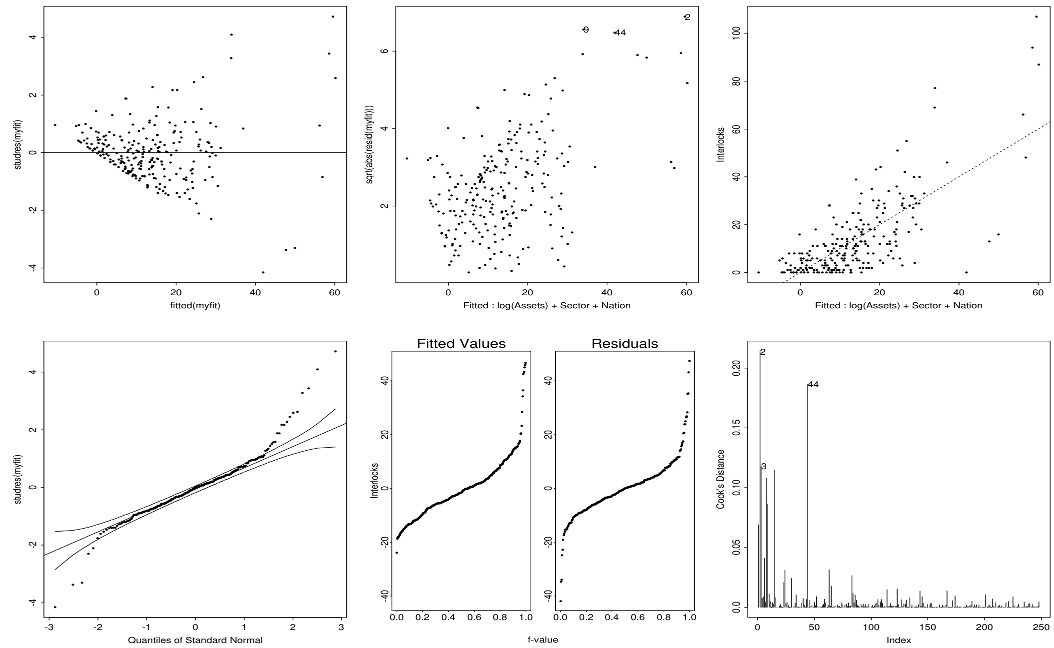

Figure 5: Modified diagnostics for the model `lm(Interlocks ~ log(Assets) + Sector + Nation)`.

```
[etc. etc. etc.]
```

There's a world of hurt in these residuals (see Figure 5)!

Both the studentized and raw residuals have dramatically non-constant variance, that is increasing with the fitted values $\hat{y}_i$. The qqnorm plot suggests very heavy tails to the left and right (consistent with the exploding residual variance in the raw residual plots). plot of the square root of the raw residuals does suggest that a square-root or similar transformation may improve things (our earlier discussion of variance-stabilizing transformations even suggests `log(y)`); and we can verify that it does (Figure 6;). The `qqnorm` plot does not suggest difficulties, but the residual plots in the first row of Figure 6 suggest there still may be some minor non-constant variance.

A different problem is the linear lower bound on the residuals as we move to the left (lower $\hat{y}_i$'s) in the plot. The source of this problem becomes clear when we look at the $\hat{y}_i$ vs y_i plot in the upper right of Figure 6: "Interlocks" is a count variable that can only take the values 0, 1, 2, etc. In particular there are quite a few companies with $y_i = 0$ and varying $\hat{y}_i$'s (probably mostly varying `Assets`); the residuals for these companies necesarily increase linearly as $\hat{y}_i$ decreases. You can see a similar linear pattern for the `Interlocks` $= 1$ companies.

2.3 Weighted Least Squares

It is also worth noting that going from Y to $\sqrt{Y}$ changes the "shape" of the regression surface: if we are in a position where there is a good substantive interpretation (ignoring `Nation` and `Sector` for the moment) for the relationship `Interlocks = a + b*log(Assets)`, we might have difficult

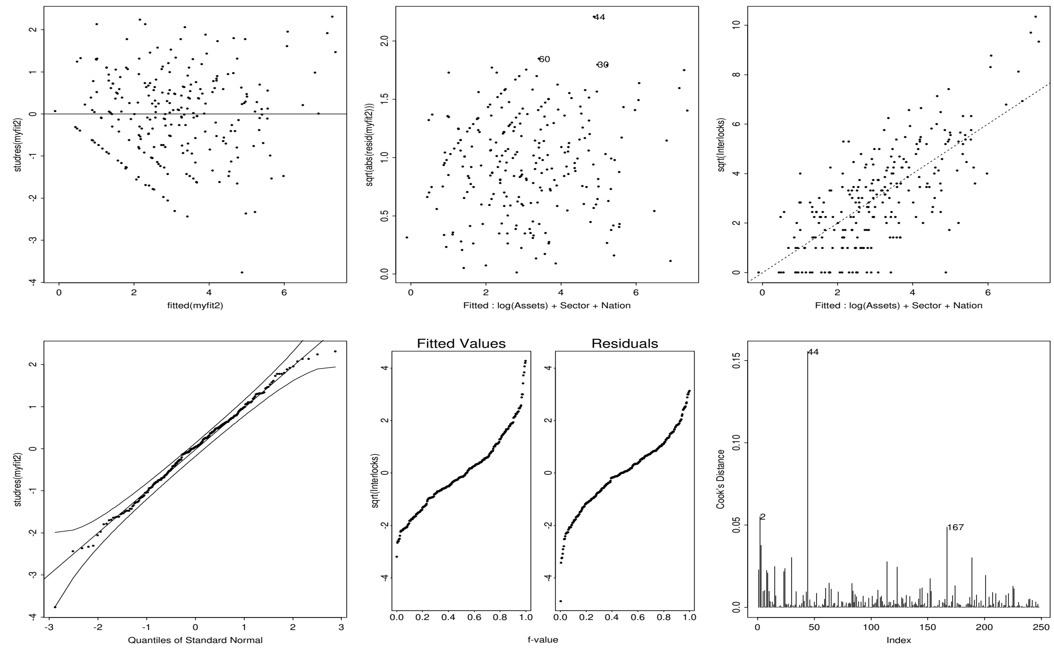

Figure 6: Modified diagnostics for the model `lm(sqrt(Interlocks) ~ log(Assets) + Sector + Nation)`.

time justifying the modified relationship `Interlocks = (a + b*log(Assets))^2`.

In cases like these, one alternative is *weighted least-squares (WLS) regression*. Suppose that the model is $\underline{y} = X\underline{\beta} + \underline{\epsilon}$ where

$$\epsilon_i \sim N(0, \sigma_\epsilon^2 / w_i^2)$$

and we know what values the weights w_i should take on. We can immediately reduce to the standard model by considering instead

$$\underline{y}^* = W^{1/2}\underline{y} = W^{1/2}X\underline{\beta} + W^{1/2}\epsilon = X^*\underline{\beta} + \epsilon^*$$

where

$$W^{1/2} = \begin{bmatrix} w_i & 0 & \cdots & 0 \\ 0 & w_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & w_n \end{bmatrix}$$

We can immediately apply our usual least-squares or normal-theory MLE machinery to show that the MLE's are

$$\begin{array}{rcccl} \hat{\underline{\beta}} & = & (X^{*T}X^*)^{-1}X^*\underline{y}^* & = & (X^TWX)^{-1}X^TW\underline{y} \\ \hat{\sigma_\epsilon^2} & = & \frac{1}{n}\sum_{i=1}^n (y_i^* - \hat{y}_i^*)^2 & = & \frac{1}{n}\sum_{i=1}^n (E_i/w_i)^2 \\ \hat{\mathrm{Var}}\,(\hat{\underline{\beta}}) & = & \hat{\sigma}_\epsilon^2 (X^{*T}X^*)^{-1} & = & \hat{\sigma}_\epsilon^2 (X^TWX)^{-1}, \quad \text{etc.} \end{array}$$

We could also obtain there results by directly differentiating the log of the likelihood

$$L(\underline{y}; \underline{\beta}, \sigma_\epsilon^2) = \frac{1}{(2\pi)^{n/2}|\underline{\Sigma}|^{1/2}} \exp\left[-\frac{1}{2}(\underline{y} - X\underline{\beta})^T \underline{\Sigma}^{-1}(\underline{y} - X\underline{\beta})\right]$$

where

$$\underset{\sim}{\Sigma} = \sigma_\epsilon^2 \begin{bmatrix} 1/w_i^2 & 0 & \cdots & 0 \\ 0 & 1/w_2^2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1/w_n^2 \end{bmatrix} = \sigma_\epsilon^2 W^{-1}$$

Either way, the answer is the same as if we had minimized the sum of weighted squared errors

$$Q = \sum_{i=1}^{n} w_i^2 E_i^2.$$

If we really have the right weights w_i, these estimates can be shown to be BLUE (Gauss-Markov theorem), etc.

In practice we have to either

- Know the weights in advance;
- Know that they are related to some observable variable; or
- Estimate the weights from the data.

I will illustrate this method by estimating the weights from our original unweighted regression, sometimes abbreviated "OLS" for "ordinary least-squares" regression. *However, this is the least likely of the three methods of obtaining weights to work well in practice,* because our WLS fits will not take into account the uncertainty in estimating the weights from the OLS regression. This uncertainty is especially felt in estimtes of the sampling variance of familiar regression objects, e.g.:

$$\hat{\mathrm{Var}}\,(\underset{\sim}{\hat{\beta}}) = \hat{\sigma}_\epsilon^2 (X^T W X)^{-1}$$

Anyway, we will forge ahead. For each data point i, we will compute w_i as the SD of the neighboring 25 (kind of arbitrary) data points in the studentized residual plot (similar results for raw residuals):

```
> ord _ order(fitted(myfit))
> resids _ studres(myfit)[ord]  # order the resids as in plot(fitted,resids)
> w _ rep(0,length(resids))
> N _ length(w)
> M _ 12
> for(i in 1:N) {
+ sel _ (i-M):(i+M)
+ while(sel[1]<1) sel _ sel + 1
+ while(sel[length(sel)]>N) sel _ sel - 1
+ w[ord[i]] _ sqrt(var(resids[sel]))   # index w in same order as resids!
+ }
```

As a check, we `plot(fitted(myfit),w)` and see a nice increasing pattern, albeit with some end effects (see Figure 7).

Fitting a weighted regression in Splus is easy(!) since `lm()` has a `weights` option that expects to see w_i^2's for WLS regression.

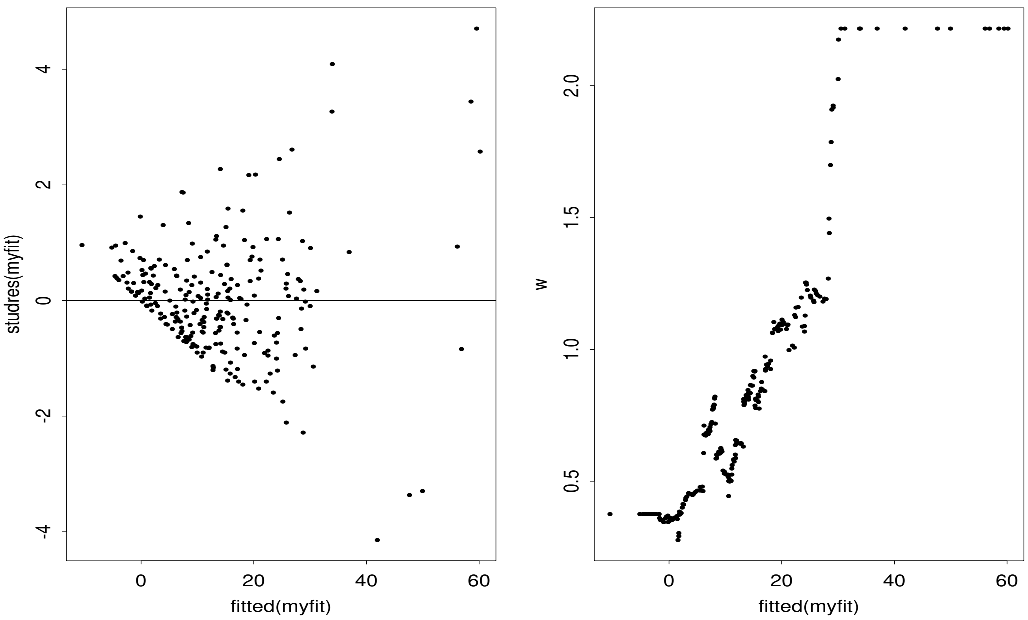

Figure 7: Studentized OLS residuals and estimated weights for applying WLS to the Interlock data.

```
707 > mywls _ lm(Interlocks ~ log(Assets) + Sector + Nation,weights=w^2)
707 > plot(mywls)
```

The resulting diagnostic plots are in Figure 8.

It might be interesting to compare the two sets of estimated coefficients, and the ANOVA tables for the two regressions; see Table 1. The main effect of using the weighted regression seems to be that the significance of the `Sector` and `Nation` factors is a bit less in the WLS regression thna in the OLS regression. This would make sense, for example, if there were great heterogeneity across nations or industry sectors in the number or variability of interlocking positions (some evidence of this exists in Figure 3 on page 5, for example).

An alternative way of obtaining weights that might be better in this problem would be to observe from the overall ANOVA for the original OLS fit (left side of Table 1) that the lion's share of the variability in the regression is driven by `log(Assets)` and to hope that some nice relationship like

$$\mathrm{Var}\,(\epsilon_i/\log(\texttt{Assets})) \approx \texttt{ constant}$$

i.e. $w_i \propto \log(\texttt{Assets})$ (this is readily verified by looking at `plot(log(Assets),residuals(myfit))`, for example). In this case, we might as well take $w_i \equiv \log(\texttt{Assets})$ in the WLS regression. (This approach works as long as we are in the conventional regression setting, in which the X's are considered to be constant).

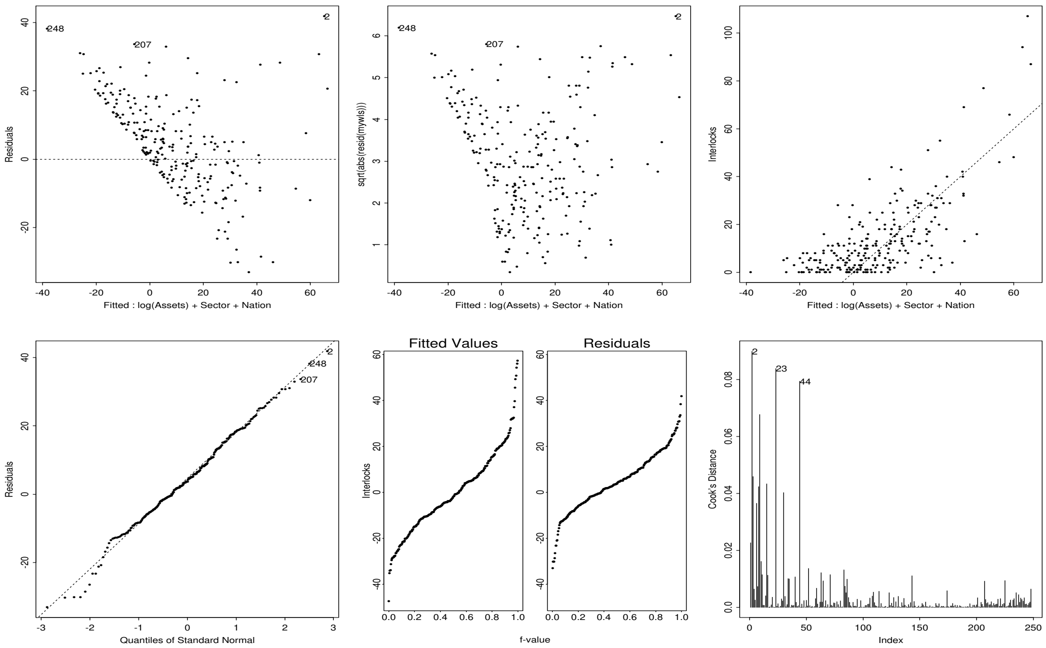

Figure 8: Standard diagnostic plots for the WLS regression `lm(formula = Interlocks ~ log(Assets) + Sector + Nation, weights = w^2)`.

```
>summary(myfit)

Call: lm(formula = Interlocks ~ log(Assets) + Sector + Nation)
Residuals:
    Min     1Q  Median    3Q   Max
 -41.94 -6.169 -0.1405 4.745 47.44

Coefficients:
                Value Std. Error  t value Pr(>|t|)
(Intercept) -30.6456   5.3897    -5.6860   0.0000
log(Assets)   5.9908   0.6814     8.7921   0.0000
    Sector1   8.6614   2.5923     3.3411   0.0010
    Sector2  -3.7914   1.9509    -1.9434   0.0532
    Sector3  -1.5361   0.8520    -1.8030   0.0727
    Sector4  -1.1251   0.9342    -1.2043   0.2297
    Sector5  -0.1608   0.4372    -0.3678   0.7133
    Sector6  -0.4183   0.4212    -0.9932   0.3216
    Sector7   0.0159   0.2670     0.0594   0.9527
    Sector8   0.0270   0.3239     0.0834   0.9336
    Sector9   0.3433   0.2820     1.2175   0.2246
    Nation1  -1.5267   1.5436    -0.9890   0.3237
    Nation2  -1.2676   1.0438    -1.2144   0.2258
    Nation3  -1.4243   0.4548    -3.1319   0.0020

Residual standard error: 11.26 on 234 degrees of freedom
Multiple R-Squared: 0.5353
F-statistic: 20.73 on 13 and 234 degrees of freedom, the p-value is 0

> anova(myfit)
Analysis of Variance Table

Response: Interlocks

Terms added sequentially (first to last)
             Df Sum of Sq  Mean Sq  F Value       Pr(F)
log(Assets)   1  28051.98 28051.98 221.0860 0.000000000
Sector        9   2981.22   331.25   2.6107 0.006855031
Nation        3   3162.63  1054.21   8.3085 0.000028366
Residuals   234  29690.55   126.88
```

```
> summary(mywls)

Call: lm(formula = Interlocks ~ log(Assets) + Sector + Nation, weights = w^2)
Residuals:
    Min    1Q Median    3Q   Max
 -66.87 -3.38  2.755 6.964 92.76

Coefficients:
                Value Std. Error  t value Pr(>|t|)
(Intercept) -78.6927   9.1020    -8.6457   0.0000
log(Assets)  11.9028   1.0845    10.9756   0.0000
    Sector1   0.7318   2.7837     0.2629   0.7929
    Sector2   0.7945   4.5506     0.1746   0.8616
    Sector3  -2.4481   1.3611    -1.7986   0.0734
    Sector4   0.9837   1.7635     0.5578   0.5775
    Sector5   1.2581   0.9026     1.3939   0.1647
    Sector6   0.1479   0.8233     0.1797   0.8576
    Sector7   0.0400   0.4759     0.0840   0.9332
    Sector8   0.4052   0.4502     0.9000   0.3691
    Sector9   0.7596   0.4276     1.7764   0.0770
    Nation1  -1.1832   2.6785    -0.4417   0.6591
    Nation2  -0.8422   2.1127    -0.3986   0.6905
    Nation3  -2.6158   0.9250    -2.8278   0.0051

Residual standard error: 16.54 on 234 degrees of freedom
Multiple R-Squared: 0.5745
F-statistic: 24.31 on 13 and 234 degrees of freedom, the p-value is 0

> anova(mywls)
Analysis of Variance Table

Response: Interlocks

Terms added sequentially (first to last)
             Df Sum of Sq  Mean Sq  F Value      Pr(F)
log(Assets)   1  77006.55 77006.55 281.6513 0.00000000
Sector        9   5397.81   599.76   2.1936 0.02327355
Nation        3   3986.76  1328.92   4.8605 0.00267598
Residuals   234  63978.16   273.41
```

Table 1: OLS and WLS fits of the Interlock regression.

3 Bootstrapping

In situations where the normal-errors assumption of linear regression models is in question, we cannot trust confidence intervals for parameters and fitted values, confidence ellipsoids, or the t- and F-distributions that are advertised for the test statistics we commonly use. *Bootstrapping* is a broadly applicable, nonparametric approach to statistical inference that uses intensive computation in place of convenient assumptions like normality. The basic idea of bootstrapping is as follows:

- Let $\mathcal{S}$ be your actual data sample, and let $\mathcal{P}$ be the population from which $\mathcal{S}$ came.
- Draw r subsamples, $\mathcal{S}_1, \ldots, \mathcal{S}_r$, from the available sample, mimicing the method by which $\mathcal{S}$ was drawn from $\mathcal{P}$ in the first place. The $\mathcal{S}_m$'s are called *bootstrap samples.*
- Whatever statistic you were interested in was some function $\hat{\theta} = g(\mathcal{S})$ of your sample. Use the same function $g()$ to calculate statistics $\hat{\theta}_1^*, \ldots, \hat{\theta}_r^*$ for each of the bootstrap (sub-)samples.
- Use properties of the empirical distribution of $\hat{\theta}_m$'s to estimate corresponding properties of the sampling distribution of $\hat{\theta}$ over repeated samples $\mathcal{S}$ from the population $\mathcal{P}$. For example, the mean of the bootstrap estimates

  $$\overline{\theta}^* = \frac{1}{r}\sum_{m=1}^{r} \hat{\theta}_m^*$$

  often exhibits less bias than $\hat{\theta}$ itself; and

  $$\widehat{SE}^*(\hat{\theta}^*) = \sqrt{\frac{1}{r-1}\sum_{m=1}^{r} (\hat{\theta}_m^* - \overline{\theta}^*)^2}$$

  is often a better estimate of the sampling variance of $\hat{\theta}$ than an asymtotic variance based on Fisher information or something similar.

Bootstrapping exploits the fundamental analogy

> The population is to the sample
> as
> the sample is to the bootstrap samples.

The amazing thing is that it works—i.e. bootstrap methods do correctly and consistently estimate sampling properties of $\hat{\theta}$'s in a wide variety of settings. Rather than prove any of this (see the books by Efron, 1982; or Efron and Tibshirani, 1993), we will apply the idea to the Interlocks regressions.

In classical regression analysis we usually imagine that the sample $\mathcal{S}$ is randomly drawn from a population of interest $\mathcal{P}$. So the r bootstrap samples should consist of n data points resampled from $\mathcal{S}$. The only question is, what is a "data point"?

- Is it the complete observation vectors $(y_i, x_{i1}, \ldots, x_{ik})$?
- Or is it just the residuals of the regression $E_i = y_i - \hat{y}_i$?

In designed experiments where the X's are clearly fixed in advance, it seems attractive to consider bootstrapping only the residuals. However, it can be shown that this approach assumes that the regression model is already specified correctly, and in particular it assumes that the regression errors ϵ_i have equal variances. So in general it is better to bootstrap the whole observation vector $(y_i, x_{i1}, \ldots, x_{ik})$.

Let's look at the original OLS regression for the `Interlocks` data, with the horrible residuals:

```
> myfit _ lm(Interlocks ~ log(Assets) + Sector + Nation)
```

I want to do two things, inspired by the comparison between the OLS and WLS fits in Table 1 above on page 11 (more is possible but this will illustrate the ideas):

1. Check the sampling distribution, bias and standard errors of the intercept and coefficient for `log(Assets)` in this regression (both changed a lot from OLS to WLS fit);
2. Since the WLS fit greatly reduced the significance of the F statistic for adding the `Sector` factor after `log(Assets)`, I want to compute a bootstrapped p-value for that test statistic.

There are a variety of built-in Splus functions to do bootstrapping (see also the Appendix of the Efron and Tibshirani, 1993, text; as well as the `S` archives on `statlib`). However, doing a bootstrap from scratch is not difficult, and shows what is going on. We use two very useful functions: `sample` to create bootstrap samples, and `update` to quickly update a linear model fit for newly-sampled data (i.e. calculate the bootstrap statistics).

```
> # Here is what sample does:
> sample(10,replace=T)
 [1]  3  1  2  2 10  6  7  3  2  2
> sort(.Last.value)
 [1]  1  2  2  2  2  3  3  6  7 10
>
> # Get the linear model initialized for use with "update"
> # and calculate the original-sample statistics we are interested in.
> N _ dim(ornstein)[[1]]
> resamp _ 1:N
> bootfit _ lm(Interlocks ~ log(Assets) + Sector + Nation,
+       data=ornstein[resamp,])
> sampstats _ c(coefficients(bootfit)[1:2])
> names(sampstats) _ c("b0","b1")
> sampstats
       b0       b1
 -30.64558 5.990782
>
> # reserve space for the bootstrap sample statistics
> r _ 20
> bootstats _ matrix(0,ncol=2,nrow=r)
> dimnames(bootstats) _ list(NULL,c("b0","b1"))
>
```

```
> # Now I would like to just say
> #
> #   for (i in 1:r) {
> #        resamp _ sample(N,replace=T)
> #        newfit _ update(bootfit)
> #        bootstats[i,] _ c(coefficients(newfit)[1:2])
> #   }
> #
> # but this can fail because the bootstrap sample might produce
> # an X matrix that is not full-rank.  So I have defined
> # "myboot()", a function for bootstrapping.  The idea is to call
> # "myboot()" as many times as needed until it stops bailing out with
> # errors from lm.  Each time it bails out it saves its work "so far".
>
> start _ 1
>
> myboot _ function() {
+   on.exit({
+       cat("Bailing on iteration",i,"...\n")
+       sample(N,replace=T) # move random number generator a bit
+       start <<- i
+       synchronize()
+   })
+   for (i in start:r) {
+       resamp _ sample(N,replace=T)
+       newfit _ update(bootfit)
+       bootstats[i,] <<- c(coefficients(newfit)[1:2])
+       synchronize()
+   }
+   on.exit()
+ }
>
> myboot()
Error in lm.fit.qr(x, y): computed fit is singular, rank 13
Bailing on iteration 2 ...
> bootstats[1:3,]
            b0       b1
[1,] -33.20998 6.356325
[2,]   0.00000 0.000000
[3,]   0.00000 0.000000
> start
[1] 2
> myboot()  # finished remaining 19 iterates successfully
```

To sample 200 bootstrap samples takes about a minute on my 90MHz Pentium PC at home. Figure 9 shows kernel density estimates and a scatter plot for `b0` and `b1`, the intercept and `log(Assets)` coefficient from simple regression. A comparison of the coefficients and their standard errors follows:

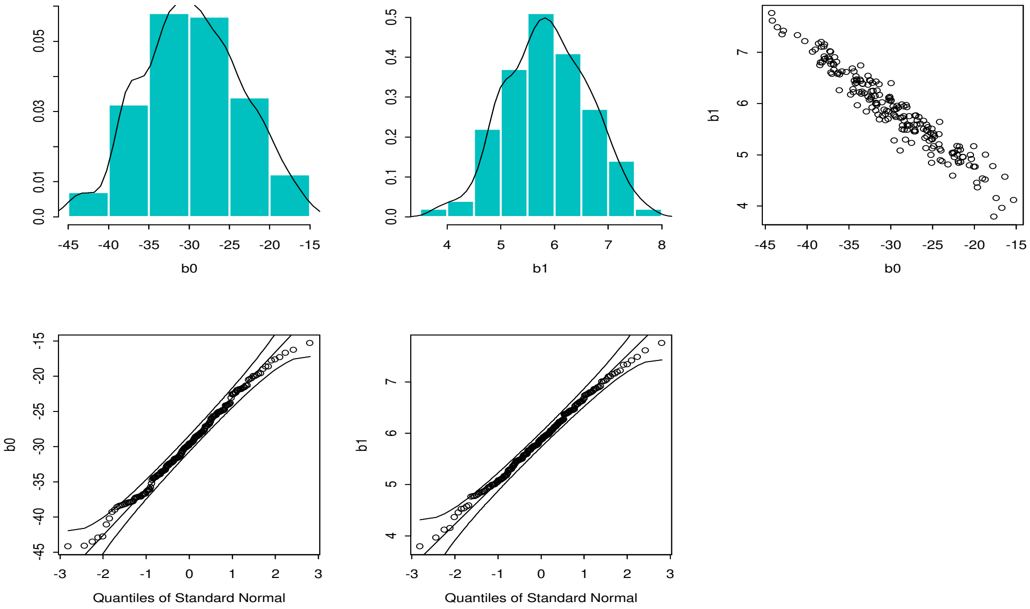

Figure 9: Bootstrap distrubutions of intercept and coefficient of `log(Assets)` in the OLS regression `lm(Interlocks ~ log(Assets) + Sector + Nation)`.

```
> par(mfrow=c(2,3))
> attach(as.data.frame(bootstats))
> hist(b0,probability=T)
> lines(density(b0,width=sqrt(var(b0))))
> hist(b1,probability=T)
> lines(density(b1,width=sqrt(var(b1))))
> plot(b0,b1)
> qqenv(b0)
> qqenv(b1)
> bootmean _ apply(bootstats,2,mean)
> bootSD _ sqrt(apply(bootstats,2,var))
> cbind(summary(bootfit)$coefficients[1:2,1:2],bootmean,bootSD)
        Value Std. Error    bootmean     bootSD
b0 -30.645577  5.3896668 -29.666767 6.1024450
b1   5.990782  0.6813797   5.867422 0.7673685
> detach()
```

The plots in Figure 9, and the comparison of OLS estimated coefficients and SE's with those of the 200-sample bootstrap, show how robust the linear regression model is to misspecification of the error distribution: as horrible as the residuals for the OLS regression were, the distribution of the $\hat{\beta}$'s is remarkably Normal. On the other hand, the scatter plot of `b0` vs `b1` has some flaring out at the lower right, suggesting that there might be violations of joint normality of the parameter estimates.

Digression.

We can compare "normal theory" confidence intervals for the parameters, with intervals from the bootstrap sample, using a larger run of 2000 bootstrap samples to obtain more stable estimates of the bootstrap percentiles. I ran this larger bootstrap while generating the plots for Figure 9 on my PC (The plot analogous to Figure 9 looks the same, but is harder to read because there is 10 times as much data).

```
> # 95% Normal theory interval for b0
> -30.645577  +c(-2,2)* 5.389666
[1] -41.42491 -19.86624
> # 95% percentile-based bootstrap interval for b0
> quantile(boot2000[,"b0"],c(0.025,0.975))
      2.5%     97.5%
 -42.88416 -17.22122

> # 95% Normal theory interval for b1
>  5.990782  +c(-2,2)*0.6813797
[1] 4.628023 7.353541
> # 95% percentile-based bootstrap interval for b1
> quantile(boot2000[,"b1"],c(0.025,0.975))
     2.5%    97.5%
 4.293508 7.579934
```

Consistent with Figure 9, the one-dimensional confidence intervals seem pretty robust to misspecification of the error distribution. (There are fancier adjustments to percentile-based bootstrap confidence intervals—one popular method is called "BC_a" by Efron and Tibshirani, 1993—but all are based on the above method.)

End of Digression.

Now the thing we really wanted to do with the larger set of 2000 bootstrap samples was generate an empirical p-value for the F statistic testing H_0 : `Sector` not needed after `log(Assets)`, vs. H_1 : `Sector` is needed.

Let $\underline{\beta}_2$ be the coefficients of the `Sector` dummy variables (`Sector is a factor!`) in the OLS regression

$$y = \beta_0 + \beta_1 \texttt{log(Assets)} + \texttt{[Dummy Matrix for Sector]}\underline{\beta}_2 + \epsilon$$

In general we want to test

$$H_0 : \ \underline{\beta}_2 = \underline{\beta}_2^0 \qquad vs \qquad H_1 : \ \underline{\beta}_2 \text{ free.}$$

Let $V_{22} = \widehat{\text{Var}}\,(\hat{\underline{\beta}}_2) = \hat{\sigma}^2_\epsilon[L(X^TX)^{-1}L^T]$ [i.e. just the rows and columns of the full matrix $\widehat{\text{Var}}\,(\hat{\underline{\beta}})$ corresponding to $\hat{\underline{\beta}}_2$]; then the F-like test statistic we are interested in is

$$T = (\underline{\beta}_2^0 - \hat{\underline{\beta}}_2)^T V_{22}^{-1} (\underline{\beta}_2^0 - \hat{\underline{\beta}}_2)$$

and the bootstrap versions of this will be

$$T_m^* = (\hat{\underline{\beta}}^*_{m2} - \hat{\underline{\beta}}_2)^T V^{*\,-1}_{m22} (\hat{\underline{\beta}}^*_{m2} - \hat{\underline{\beta}}_2) \ , \quad m = 1, \ \ldots, \ r$$

Note that the logic here is somewhat different from the logic for F-testing:

- In traditional F-testing $\hat{\underline{\beta}}_2$ is considered random and we calculate the distribution of T under under H_0; H_0 is rejected when the observed T is unusually large relative to this "null" distribution.
- In bootstrap testing as outlined above, both $\underline{\beta}_2^0$ and $\hat{\underline{\beta}}_2$ are considered fixed (by H_0 and by the observed data, respectively), and hence T is fixed also, when we are generating bootstrap values for T_m^*. The distribution of the T_m^*, which are estimating 0 no matter what H_0 says, is used as a gauge to see whether T is large (reject H_0) or small (do not reject H_0).

Hence, The bootstrap p-value for the test of H_0 vs. H_1 is

$$p = \frac{1}{r} \sum_{m=1}^{r} 1_{\{T_m^* > T\}}$$

The basic code to get T and the T^*'s is listed below:

```
> resamp _ 1:N
> bootfit _ lm(Interlocks ~ log(Assets) + Sector,data=ornstein[resamp,])
> Fcomp _ function(fit) {
+ # read help(summary.lm)!!
+   fit _ summary(fit)
+   vci _ solve((fit$cov.unscaled * fit$sigma^2)[-c(1,2),-c(1,2)])
+   delta _ coefficients(fit)[-c(1,2),1] - betas
+   return( delta %*% vci %*% delta )
+ }
>
> # compute T
> betas _ rep(0,length(coefficients(bootfit))-2)
> Fstat0 _ Fcomp(bootfit)
> Fstat0
        [,1]
[1,] 21.5063
>
> # set things up for the T^*'s
> betas _ coefficients(bootfit)[-c(1,2)]
> r _ 20
> Fstat _ rep(0,r) # To hold T^*'s
>
> start _ 1
>
> myboot _ function() {
+  on.exit({
```

```
+       cat("Bailing on iteration",i,"...\n")
+       sample(N,replace=T) # move random number generator a bit
+       start <<- i
+       synchronize()
+   })
+   for (i in start:r) {
+       resamp _ sample(N,replace=T)
+       newfit _ update(bootfit)
+       Fstat[i] <<- Fcomp(newfit)
+       synchronize()
+   }
+   on.exit()
+ }
>
> myboot()
Error in lm.fit.qr(x, y): computed fit is singular, rank 10
Dumped
Bailing on iteration 16 ...
> myboot() # finished remaining iterations
```

To have a good handle on small p-values, we need many many samples. We will try to do this with 2000 bootstrap samples, which takes a while to generate. Once we have them, we simply have to plot the distribution of the 2000 T^*_m statistics, and see how large the upper tail is that the original T-value of 21.5063 cuts off.

```
> hist(Fstat,probability=T)
> lines(density(Fstat,width=sqrt(var(Fstat))))
> abline(v=21.5063)
> mean(Fstat>21.5063)
[1] 0.177
```

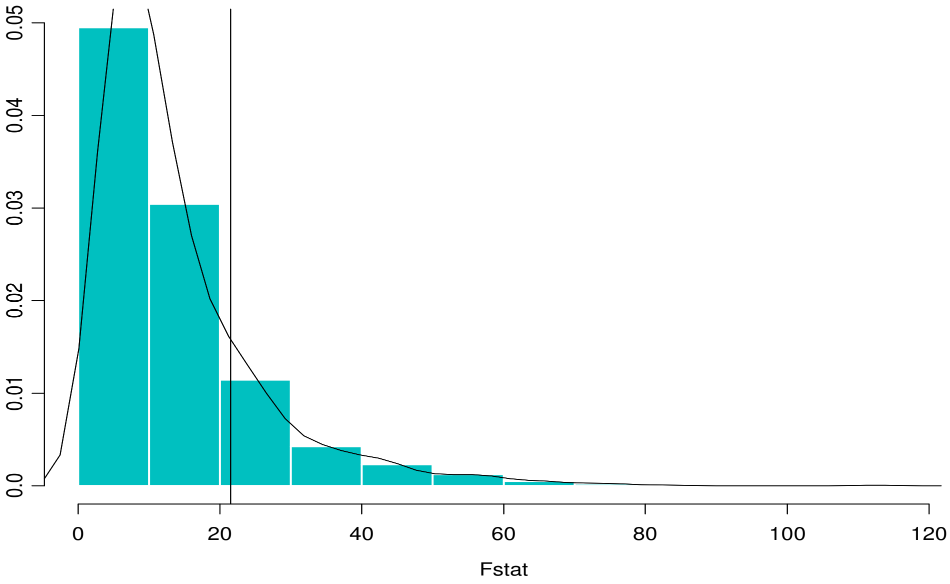

Figure 10: Bootstrap test of the hypothesis that `Sector` is not needed after fitting `log(Assets)`, in the OLS regression `lm(Interlocks ~ log(Assets) + Sector )`. The vertical line, at 21.5063, cuts off an upper tail accounting for 17.7% of the bootstrap distribution.

The p-value is quite high, 0.177, vs. the uncorrected OLS p-value of 0.006855031 and the WLS p-value of 0.02327355, given in the anova tables listed in Table 1. Taking the bootstrap as the gold standard here, we have seen that

- OLS can be quite robust to mis-specification of the error distribution, especially for simple inferences on single parameters;
- OLS can be wildly wrong in F-testing when the error distribution is mis-specified;
- WLS is an improvement over OLS for F-testing, but still can be "way off".