Model and Data Issues

Statistics and Econometrics

Jiahua Wu

Part I

Functional form misspecification

The R function for RESET (on slide 12) is *resettest*(), which is readily available from the *lmtest* package. It is based on an F test for the expanded model in Step 2 with null hypothesis $H_0 : \delta_{\hat{y}^2} = 0, \delta_{\hat{y}^3} = 0$. If we reject null, we reject the null hypothesis that functional form is correctly specified. Smaller the p-value, stronger the evidence against null.

```
load("hprice1.RData")

# RESET test
house.m1 <- lm(price ~ lotsize + sqrft + bdrms, data)
resettest(house.m1, type = "fitted")
```

```
##
##  RESET test
##
## data:  house.m1
## RESET = 4.6682, df1 = 2, df2 = 82, p-value = 0.01202
```

Judging by the p-value, there is evidence of functional form misspecification in the model. As we mentioned in the class, one problem with RESET test is that it does not provide any direction on how to adjust the model when the null hypothesis is rejected. We will need to rely on theory/common sense/subject knowledge to guide the model construction.

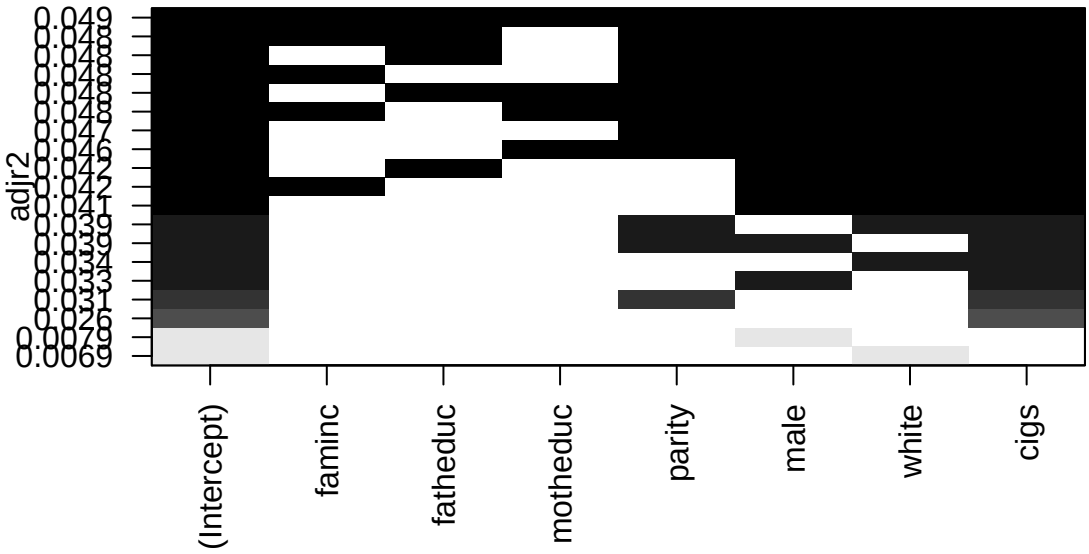
Variables Selection

In the class, we discussed two functions that will automatically evaluate different models, and return the best ones depending on goodness-of-fit measure of choice. The first function is *regsubsets*() from *leaps* package. It is based on an exhaustive search, and thus computational complexity increases exponentionally (in the number of independent variables).

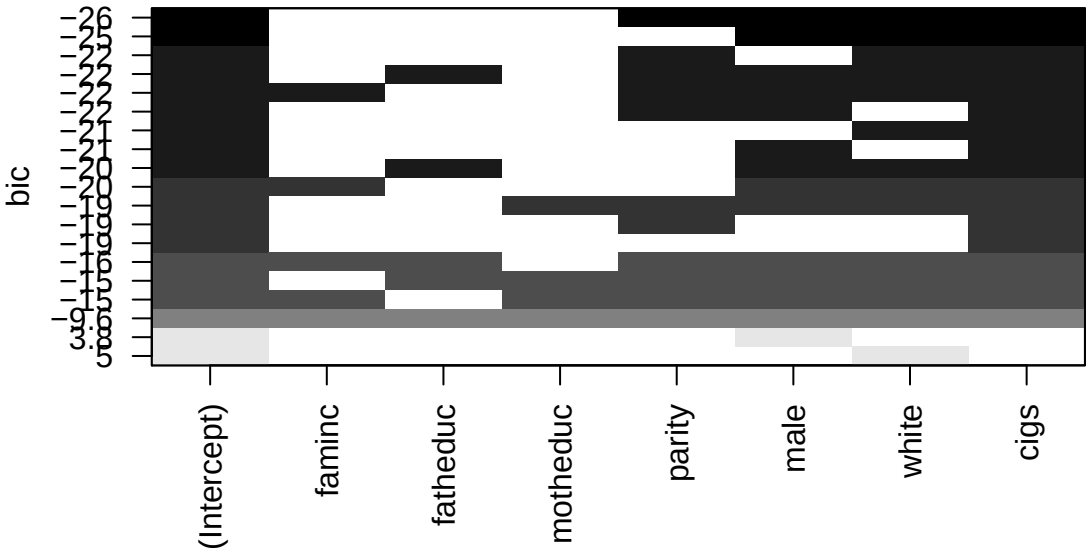
```
load("bwght.RData")
data.new <- na.omit(data)
bwght.model.search <- regsubsets(bwght ~ faminc + fatheduc + motheduc + parity
                                 + male + white + cigs, data.new, nbest = 3)
```

Before running *regsubsets*(), we first remove any observations with missing values in the sample, such that different models are estimated with the same sample. *nbest* specifies the number of subsets of each size to record. We plot the models based on two goodness-of-fit measures, namely $\bar{R}^2$ and BIC.

```
plot(bwght.model.search, scale = "adjr2")
```




```
plot(bwght.model.search, scale = "bic")
```



The way to read the figures is as follows: each row indicates one model and each column indicates one independent variable. If a block is black (or grey, they are the same), it indicates that the corresponding independent variable (column) is included in the corresponding model (row). One thing worth-mentioning is that the vertical axis in the BIC figure indicates the difference of BICs between that particular model and a model with only the intercept, rather than the model BIC itself.

The other function we discussed for automatic variable selection is *step*(). It would perform stepwise search to find the model with the best goodness-of-fit measure. The function can search $forward$, *backward*, or in *both* directions. With $forward$ search, at each iteration, the function would evaluate the model AICs by adding independent variables that are not already included in the model one at a time, and add the one with the lowest AIC at the end of the iteration. The algorithm stops when no more independent variables can be added to the model for a lower AIC. With *backward* search, at each iteration, the function would evaluate the model AICs by removing independent variables that are already in the model one at a time, and remove the one such that the resulting model has the lowest AIC at the end of the iteration.

In the end, *step*() will return one model with the lowest AIC. As discussed in the class, we may find several models with similar AICs (with difference less than 2), then we can decide whether to include those variables (in one model but not the other) based on subject knowledge, and etc. There is no fixed rule here. Model validation (which you will learn in the machine learning class) could also help you to pick the model.

```
# stepwise search
bwght.null <- lm(bwght ~ 1, data.new)
bwght.full <- lm(bwght ~ faminc + fatheduc + motheduc + parity + male + white
                 + cigs, data.new)
step(bwght.null, scope = list(lower = bwght.null, upper = bwght.full),
     direction = "forward")
step(bwght.full, direction = "backward")
step(bwght.null, scope = list(lower = bwght.null, upper = bwght.full),
```

```
     direction = "both")
```

If we want to evaluate models based on BIC, we can simply add the argument $k = log(n)$ in the $step()$ function.

```
n <- nrow(data.new)
step(bwght.null, scope = list(lower = bwght.null, upper = bwght.full),
     direction = "forward", k = log(n))
```

Prediction

The function for model prediction in R is $predict()$. One argument that is worth mentioning is $interval$. If we specify $interval =$ "$confidence$", then we are constructing a confidence interval for the average person - error term u does not play a role here as on average it is equal to 0. On the other hand, if we specify $interval =$ "$predict$", we are constructing an interval prediction for a particular individual. This is often called "prediction interval". For prediction interval, we need to account for two sources of variations: sampling variation (as we don't really know the true population parameter), and variance in the error term (as we don't observe it for this particular individual). As such, the prediction interval would be much larger.

```
load("wage1.RData")
wage.m1 <- lm(wage ~ educ, data)
newdata <- data.frame(educ = 12)

# prediction for an average person
predict(wage.m1, newdata, interval = "confidence", level = 0.95)
```

```
##        fit      lwr      upr
## 1 5.591459 5.296152 5.886766
```

```
# prediction for a particular individual
predict(wage.m1, newdata, interval = "predict", level = 0.95)
```

```
##        fit       lwr      upr
## 1 5.591459 -1.051958 12.23488
```

Lastly, if we want to predict y in a model where the dependent variable is $log(y)$, simply exponentiating the fitted value wouldn't give us the right answer. We need to scale it up by a sample estimate of $E(exp(u))$, which can be estimated by calculating the sample average of exponential of residuals.

```
wage.m2 <- lm(log(wage) ~ educ, data)
predicted.logwage <- predict(wage.m2, newdata, interval = "none")
predicted.wage <- mean(exp(wage.m2$residuals)) * exp(predicted.logwage)
```

Part II

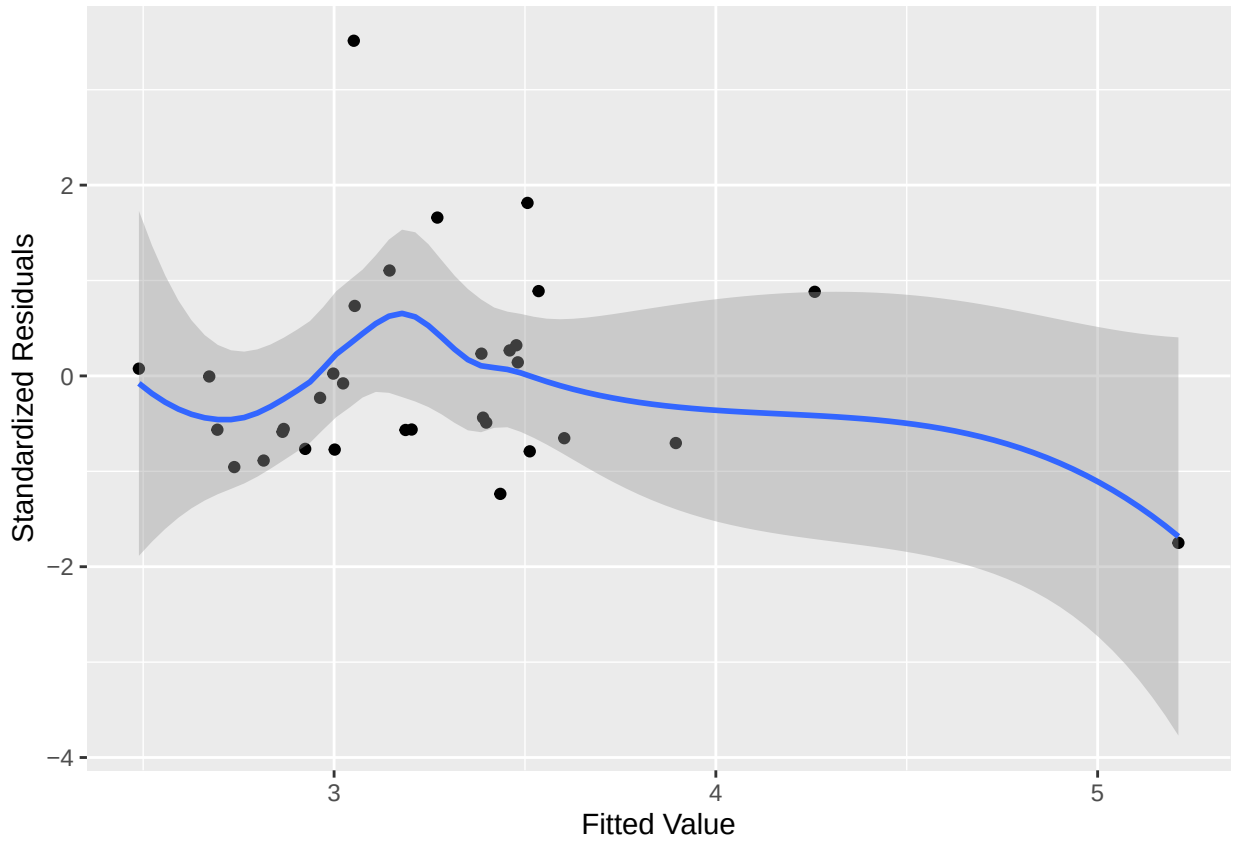
Outlier and leverage points

One common way to identify outliers/leverage points in the sample is through visual inspection. Plots that involve standardized residuals, leverage values or Cook's distances can all help us with this regard. We draw two plots for this particular example. The first one is "standardized residuals vs fitted values", which can help us identify outliers (unusual y values). The rule of thumb is an observation is considered as an outlier if it lies 3 standard deviations away from the mean. Judging from the plot, we have one outlier in this example, just above 3 standard deviations from the mean.

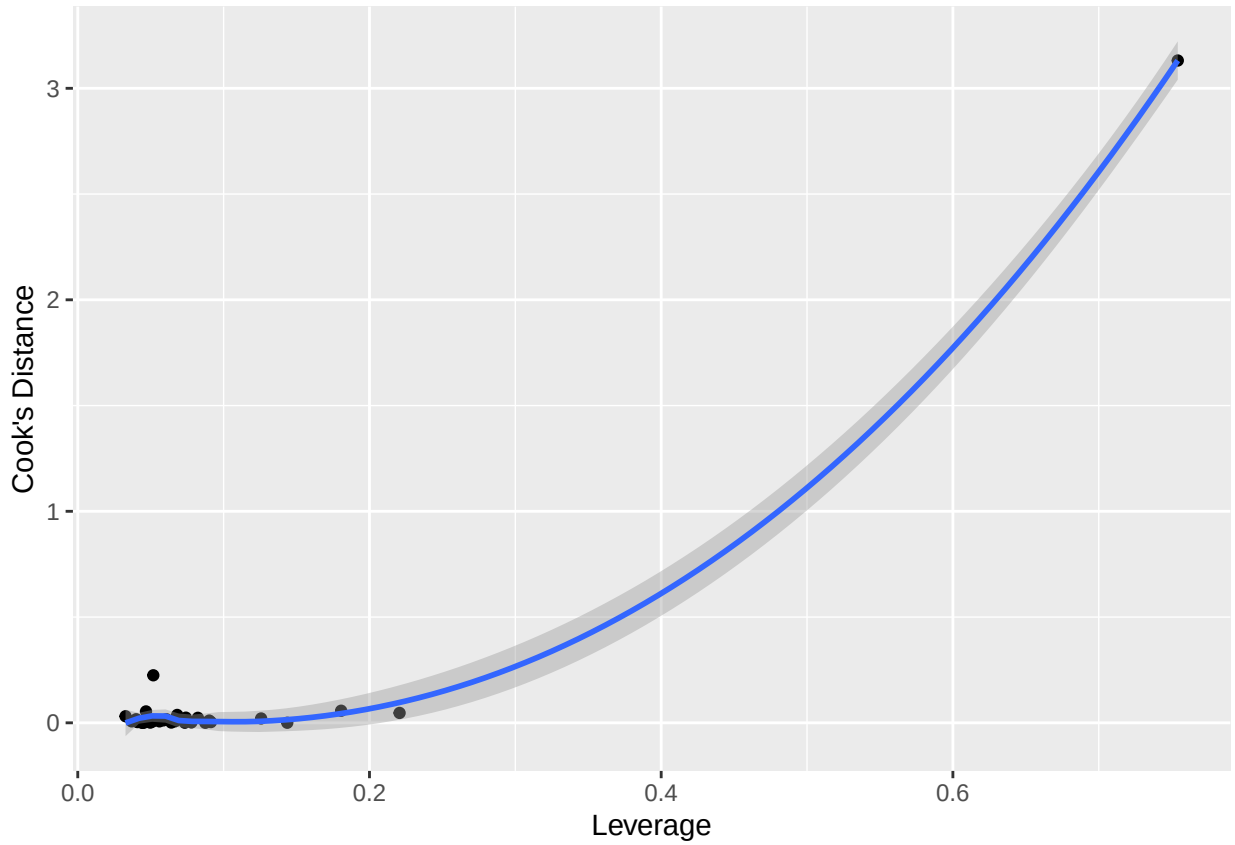
The second plot is "Cook's distance vs leverage values", which can help us identify high leverage points (unusual x values), and influential observations. An observation has high leverage if its leverage value is greater than twice of the mean $(2(k+1)/n)$. Influential observations are those ones with Cook's distance greater than 1.

```
load("rdchem.RData")
rdchem.m1 <- lm(rdintens ~ sales + profmarg, data)

ggplot(rdchem.m1, aes(.fitted, .stdresid)) + geom_point() + stat_smooth(method = "loess") +
  xlab("Fitted Value") + ylab("Standardized Residuals")
```




```
ggplot(rdchem.m1, aes(.hat, .cooksd)) + geom_point() + stat_smooth(method = "loess") +
  xlab("Leverage") + ylab("Cook's Distance")
```



In the example, we have one influential observation with Cook's distance around 3.13, which implies that our OLS regression function would change significantly with or without this observation.

```
# standardized residual
rstandard(rdchem.m1)
```

```
##           1           2           3           4           5
##  3.513263291 -0.790519848  0.267128732  0.076006890  0.321280905
##           6           7           8           9          10
## -0.566687483 -0.005976001  0.733693535  0.881712154 -1.749820825
##          11          12          13          14          15
## -0.764851623 -0.437421719 -0.586030540 -0.563911976  0.024206213
##          16          17          18          19          20
## -0.561927126 -0.703683414 -0.078196355 -0.652626190  0.233240569
##          21          22          23          24          25
##  0.889011423  1.813625172  0.142520660 -1.236267179 -0.229724083
##          26          27          28          29          30
##  1.105018054 -0.555999465 -0.887124280  1.659851483 -0.955445027
##          31          32
## -0.489346764 -0.772089271
```

```
# leverage value
hatvalues(rdchem.m1)
```

```
##          1          2          3          4          5          6
## 0.05178072 0.05917460 0.07793055 0.14364715 0.09129965 0.04097883
##          7          8          9         10         11         12
## 0.08744683 0.03698317 0.18058297 0.75418395 0.05875135 0.04359415
```

```
##         13         14         15         16         17         18
## 0.05599620 0.09012704 0.04412758 0.05115140 0.22063132 0.04518525
##         19         20         21         22         23         24
## 0.12571730 0.06412486 0.08234326 0.04678718 0.07338371 0.06816198
##         25         26         27         28         29         30
## 0.04968940 0.04001925 0.06678673 0.06077687 0.03275102 0.07411315
##         31         32
## 0.04100128 0.04077131
```

```
# cooks distance
cooks.distance(rdchem.m1)
```

```
##            1            2            3            4            5
## 2.246774e-01 1.310179e-02 2.010315e-03 3.230196e-04 3.456981e-03
##            6            7            8            9           10
## 4.574012e-03 1.140738e-06 6.890940e-03 5.710895e-02 3.131360e+00
##           11           12           13           14           15
## 1.217157e-02 2.907136e-03 6.790536e-03 1.049967e-02 9.016596e-06
##           16           17           18           19           20
## 5.674131e-03 4.672589e-02 9.645603e-05 2.041507e-02 1.242497e-03
##           21           22           23           24           25
## 2.363966e-02 5.381593e-02 5.362091e-04 3.726535e-02 9.197927e-04
##           26           27           28           29           30
## 1.696774e-02 7.374571e-03 1.697529e-02 3.109594e-02 2.435721e-02
##           31           32
## 3.412649e-03 8.445907e-03
```

Judging by the Cook's distance, the influential observation is observation 10. So if we drop this observation and run regression again, firm's size now has a significant impact on R&D expenditures.

```
data.new <- data[-c(10), ]
rdchem.m2 <- lm(rdintens ~ sales + profmarg, data.new)
summary(rdchem.m2)
```

```
##
## Call:
## lm(formula = rdintens ~ sales + profmarg, data = data.new)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.0687 -1.1867 -0.7956  0.6486  6.0811
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 2.2968508  0.5918045   3.881 0.000577 ***
## sales       0.0001856  0.0000842   2.204 0.035883 *
## profmarg    0.0478411  0.0444831   1.075 0.291336
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.792 on 28 degrees of freedom
## Multiple R-squared:  0.1728, Adjusted R-squared:  0.1137
## F-statistic: 2.925 on 2 and 28 DF,  p-value: 0.07022
```