Stat 104: Quantitative Methods for Economists

Class 35: Multiple Regression

Residuals and Their Plots

All of the assumptions of the model are really statements about the regression error terms (ε)

How can we test whether the data supports these assumptions if we cannot observe the errors directly? We rely on diagnostics that use basic *least squares residuals* $e_i = Y_i - \hat{Y}_i$

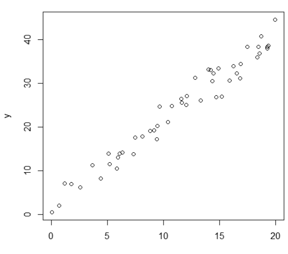
We pretend as though the least squares residuals are the same as the true regression errors… with some limitations.
Sometimes we use **Standardized Residuals** for convenience.

$$\boxed{r_i = \frac{e_i}{s_e} \approx \frac{\varepsilon_i}{\sigma} \sim N(0,1)}$$

(why are these useful ?)

When things are right

Consider the data:



this plot looks like the kind of data our model is meant to describe.

Always plot Y vs X!

As a further check we examine the residuals.

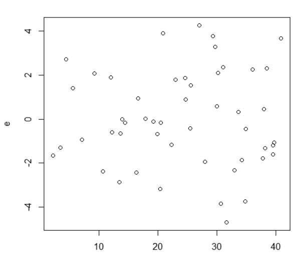
Obtaining Residuals in R

- We need the residuals, fitted values and standardized residuals

```
> fit=lm(y~x)
> e=residuals(fit)
> yhat=fitted(fit)
> sres=rstudent(fit)
```


Plot residuals versus Yhat

```
plot(yhat,e)
```



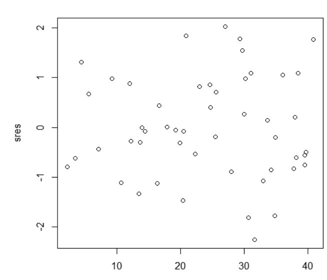
This is the way a residual plot looks when the model fits the data:

No obvious pattern!!!!!

resids unrelated to X!!!!!!

$$Y = \hat{Y} + e$$

(or) Plot standardized residuals vs Yhat



no obvious pattern!!!!!
resids unrelated to X!!!!!!
standardized resids between -2 and +2!!!!!!

Normality of Error Terms

- A major, big-time assumption is that the errors in our regression model are normally distributed. $\varepsilon \sim N(0,1)$
- This assumption lets us construct confidence intervals and do hypothesis tests.
- It is essential that we always check this assumption

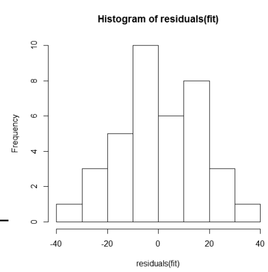
Normality Tests in R

- The null hypothesis of each test is “data is normally distributed”
- R package `nortest`
- `ad.test(residuals(fit))`

Ho: Normal
Ha: Not Normal
We want a high p-value

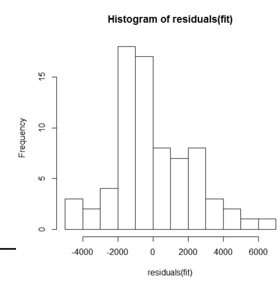
Crying Baby Data

```
> fit=lm(IQ~Crying,data=foo)
> ad.test(residuals(fit))

        Anderson-Darling normality test

data:  residuals(fit)
A = 0.24283, p-value = 0.7507
```



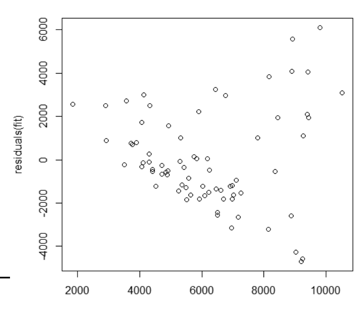
Car price model regression

```
> fit=lm(price~mpg+weight+length+turn+headroom,data=foo)
> ad.test(residuals(fit))

        Anderson-Darling normality test

data:  residuals(fit)
A = 0.82337, p-value = 0.03195
```



Check residual plot

- Hmmmm-we’ll see this in just a minute what this plot means.



What if it is not normal?

- In cases in which the normality assumption is not satisfied, transforming the dependent variable is often useful.
- In many instances, a log transformation works
- Also, the presence of outliers can distort the results of the normality test.

Multiple Regression

- Multiple Regression allows us to:
  - Use several variables at once to explain the variation in a continuous dependent variable.
  - Isolate the unique effect of one variable on the continuous dependent variable while taking into consideration that other variables are affecting it too.
  - Write a mathematical equation that tells us the overall effects of several variables together and the unique effects of each on a continuous dependent variable.

The Multiple Regression Model

- Multiple linear regression is very similar to simple linear regression except that the dependent variable Y is described by k independent variables $X_1, \ldots, X_k$

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_k X_k + \varepsilon$$

Multiple Linear Regression

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_k X_k + \varepsilon$$

- Intercept is the same
- Slope b_i is the change in Y given a unit change in X_i while holding all other variables constant (more on this later)
- SST, SSE, SSR, and R^2 are the same
- s_e is the same except now s_e = sqrt(SSE / (n-k-1))
- Slope coefficient C.I.s are the same
- p-values (one for each X_i) are the same

Example : Housing Data

We have data on 15 randomly selected house sales from last year:

price	size	age	lotsize
89.5	20.0	5	4.1
79.9	14.8	10	6.8
83.1	20.5	8	6.3
56.9	12.5	7	5.1
66.6	18.0	8	4.2
82.5	14.3	12	8.6
126.3	27.5	1	4.9
79.3	16.5	10	6.2
119.9	24.3	2	7.5
87.6	20.2	8	5.1
112.6	22.0	7	6.3
120.8	19.0	11	12.9
78.5	12.3	16	9.6
74.3	14.0	12	5.7
74.8	16.7	13	4.8

price in $1000's

size in 100 sq-feet

age in years

lot size in 1000 sq-feet

How does selling price relate to the three variables ?

```
> fit=lm(price~size+age+lotsize)
> summary(fit)

Call:
lm(formula = price ~ size + age + lotsize)

Residuals:
     Min       1Q   Median       3Q      Max
-14.3848  -1.7477   0.5549   4.0566   8.6598

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -16.0580    19.0710  -0.842 0.417712
size          4.1462     0.7512   5.520 0.000181 ***
age          -0.2361     0.8812  -0.268 0.793730
lotsize       4.8309     0.9011   5.361 0.000230 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 6.894 on 11 degrees of freedom
Multiple R-squared:  0.9161,    Adjusted R-squared:  0.8932
F-statistic: 40.03 on 3 and 11 DF,  p-value: 3.278e-06
```

s_e

Interpretation:

The relationship between house size and price is measured by b_1 = 4.146. This indicates that in this model, for each additional 100 square feet, the price of the house increases (on average) by $4,146 (assuming that the other independent variables are fixed).

The coefficient b_2 = -.236 specifies that for each additional year in the age of the house, the price decreases by an average of $236 (as long as the values of the other independent variables do not change).

The coefficient b_3 = 4.831 means that for each additional 1000 sq-feet if lot size, the price increases by an average of $4831 (assuming that house size and age remain the same).

This Held Fixed Concept

- In a multiple regression model, the interpretation of a parameter is entirely dependent upon the model in which the parameter appears.
- If you have the "wrong" sign, you may not be thinking clearly about the "held fixed" meaning of the parameters (it can be confusing).

Example

- Consider data where Y = SAT score, X_1 = High School GPA, X_2 = Time (1992 - 2002).

```
SAT     |                  3    4 4
        |                   3      4   4
        |      1        2  3  3          4
        |               2  2        4 3
        |     1    1    1  2          3
        |              1       22
        |        1 1
        |       1      1
         ------------------------------------------
            1990            2000             Time
```

(Symbol plotted is value of X_1, in Grade points: 1=D, 4=A).

Example

- While SATs are generally increasing over time, the SATs are decreasing within each grade strata, as evidenced by the decreasing pattern within each of GPAs 1,2,3,4.
- In the case of Rinott and Tam's study, they argued that this discrepancy is caused by grade inflation.

```
SAT     |                  3    4 4
        |                   3      4   4
        |      1        2  3  3          4
        |               2  2        4 3
        |     1    1    1  2          3
        |              1       22
        |        1 1
        |       1      1
         ------------------------------------------
            1990            2000             Time
```


Example

- The sign of the estimate of b_2 in the multiple regression model SAT = b_0 + b_1GPA + b_2Time + e, will be negative, and might seem "wrong", but it is actually correct when you think about the "held fixed" meaning (specifically, holding GPA fixed).
- In other words, the "partial" relationship between SAT and Time is a decreasing relationship

Example

- On the other hand, the sign of the estimate of b_1 in the simple regression model SAT = b_0 + b_1Time + e, will be positive, reflecting the generally increasing trend.
- "Simpson's paradox" refers to the reversal of signs of directional associations that sometimes occurs when data are aggregated. Here, in the GPA-defined subgroups, we see negative trends. However, in the aggregate data, we see a positive trend.

R-squared

ho hum as before,

$$SST = SSR + SSE$$

$$R^2 = \frac{SSR}{SST}$$

Confidence Intervals and Hypothesis Tests

confidence intervals are as before:

$$b_j \pm 1.96 s_{b_j}$$

and the hypothesis test:

$$\text{reject } H_0 : \beta_j = \beta_j^* \quad \text{if}$$

$$t = \left| \frac{b_j - \beta_j^*}{s_{b_j}} \right| \geq 1.96$$

Or as always reject the null if the P-value < 0.05

The housing data :

	price	Coef.	Std. Err.	t	P>|t|
b_1	size	4.146191	.7511855	5.52	0.000
b_2	age	-.2360837	.8812207	-0.27	0.794
b_3	lotsize	4.830881	.901075	5.36	0.000
	_cons	-16.05802	19.07105	-0.84	0.418

Clearly we “accept” $H_0 : \beta_0 = 0$, and $H_0 : \beta_2 = 0$

Clearly we reject $H_0 : \beta_1 = 0$, and $H_0 : \beta_3 = 0$

A confidence interval for β_1 is:

4.145 +/- 1.96(.751)

Example : Is brain and body size predictive of intelligence?

- Sample of n = 38 college students
- Response (Y): intelligence based on **PIQ** (performance) scores from the (revised) Wechsler Adult Intelligence Scale.
- Potential predictor (x_1): Brain size based on **MRI** scans (given as count/10,000).
- Potential predictor (x_2): **Height** in inches.
- Potential predictor (x_3): **Weight** in pounds.


```
> fit=lm(piq~brain+height+weight)
> summary(fit)

Call:
lm(formula = piq ~ brain + height + weight)

Residuals:
   Min     1Q Median     3Q    Max
-32.73 -12.09  -3.84  14.17  51.69

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) 111.378186  62.971483    1.77  0.08591 .
brain         2.060200   0.563455    3.66  0.00086 ***
height       -2.732402   1.229522   -2.22  0.03302 *
weight        0.000716   0.197064    0.00  0.99712
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 20 on 34 degrees of freedom
Multiple R-squared:  0.295,     Adjusted R-squared:  0.233
F-statistic: 4.74 on 3 and 34 DF,  p-value: 0.00722
```

Interpretation ?

The IQ Data again:

```
> fit=lm(piq~brain+height+weight)
> summary(fit)

Call:
lm(formula = piq ~ brain + height + weight)

Residuals:
   Min     1Q Median     3Q    Max
-32.73 -12.09  -3.84  14.17  51.69

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) 111.378186  62.971483    1.77  0.08591 .
brain         2.060200   0.563455    3.66  0.00086 ***
height       -2.732402   1.229522   -2.22  0.03302 *
weight        0.000716   0.197064    0.00  0.99712
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 20 on 34 degrees of freedom
Multiple R-squared:  0.295,     Adjusted R-squared:  0.233
F-statistic: 4.74 on 3 and 34 DF,  p-value: 0.00722
```

```
> confint(fit)
              2.5 % 97.5 %
(Intercept) -16.60 239.35
brain         0.92   3.21
height       -5.23  -0.23
weight       -0.40   0.40
```

Clearly we “accept” $H_0 : \beta_0 = 0$, and $H_0 : \beta_3 = 0$

Clearly we reject $H_0 : \beta_1 = 0$, and $H_0 : \beta_2 = 0$

Adjusted R-squared

It can be shown that every time you add a new X variable to a multiple regression the error sum of squares (SSE) goes down (math fact).

Since,

$$R^2 = 1 - \frac{\sum_{i=1}^{n} e_i^2}{\sum_{i=1}^{n} (Y_i - \bar{Y})^2} = 1 - \frac{SSE}{SST}$$

this means that every time you add an X variable, R^2 goes up.

People love R^2.

This leads to an overwhelming temptation to put lots of X's in.

This is a bad attitude. We want to summarize and predict, and we want to to it in the simplest possible way. The more complicated a model is, the less use it tends to be. Of course it has to be "complicated enough" to capture the important features of the data.

The adjusted R-squared is designed to build in an automatic penalty for adding an X.

$$R_a^2 = 1 - \frac{\frac{1}{n-k-1}\sum_{i=1}^{n} e_i^2}{\frac{1}{n-1}\sum_{i=1}^{n}(Y_i - \bar{Y})^2} = 1 - \frac{\frac{1}{n-k-1}SSE}{\frac{1}{n-1}SST}$$

I find the "penalty" artificial.

Example

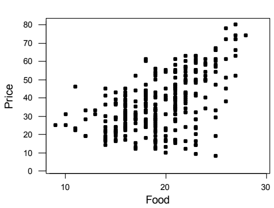
- Note the difference between regular and adjusted R-sq

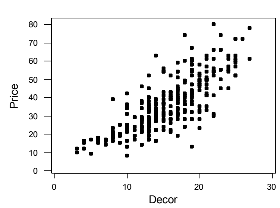
```
> fit=lm(piq~brain+height+weight)
> summary(fit)

Call:
lm(formula = piq ~ brain + height + weight)

Residuals:
   Min     1Q Median     3Q    Max
-32.73 -12.09  -3.84  14.17  51.69

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept) 111.378186  62.971483    1.77  0.08591 .
brain         2.060200   0.563455    3.66  0.00086 ***
height       -2.732402   1.229522   -2.22  0.03302 *
weight        0.000716   0.197064    0.00  0.99712
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 20 on 34 degrees of freedom
Multiple R-squared:  0.295,     Adjusted R-squared:  0.233
F-statistic: 4.74 on 3 and 34 DF,  p-value: 0.00722
```

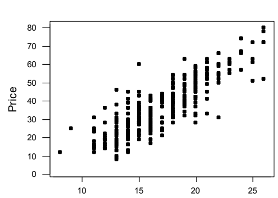

Another Example

- Consider Zagat food ratings for Manhattan
- We have data on price of meal, and ratings for food quality, décor and service.

Zagat data : Relationship between price and other variables at the same time!







First,

- Regress price on food quality:

```
> fit=lm(price~food)
> summary(fit)

Call:
lm(formula = price ~ food)

Residuals:
   Min     1Q Median     3Q    Max
-23.49  -8.31  -1.85   7.11  42.59

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   -3.871     10.047   -0.39  0.70046
food           1.640      0.436    3.76  0.00022 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 12 on 193 degrees of freedom
Multiple R-squared:  0.0683,    Adjusted R-squared:  0.0635
F-statistic: 14.2 on 1 and 193 DF,  p-value: 0.000223
```

Now a multiple regression

```
> fit=lm(price~food+decor+service)
> summary(fit)

Call:
lm(formula = price ~ food + decor + service)

Residuals:
   Min     1Q Median     3Q    Max
-20.50  -5.90  -0.38   4.78  47.76

Coefficients:
            Estimate Std. Error t value    Pr(>|t|)
(Intercept) -28.3326     8.2553   -3.43    0.00073 ***
food         -0.0401     0.3993   -0.10    0.92016
decor         0.7471     0.2702    2.76    0.00626 **
service       2.5097     0.4495    5.58 0.00000008 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 9.7 on 191 degrees of freedom
Multiple R-squared:  0.424,     Adjusted R-squared:  0.415
F-statistic: 46.8 on 3 and 191 DF,  p-value: <0.0000000000000002
```


Compare: What Happened?

```
Call:
lm(formula = price ~ food)

Residuals:
   Min     1Q Median     3Q    Max
-23.49  -8.31  -1.85   7.11  42.59

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   -3.871     10.047   -0.39  0.70046
food           1.640      0.436    3.76  0.00022 ***
---

Call:
lm(formula = price ~ food + decor + service)


Coefficients:
            Estimate Std. Error t value    Pr(>|t|)
(Intercept) -28.3326     8.2553   -3.43    0.00073 ***
food         -0.0401     0.3993   -0.10    0.92016
decor         0.7471     0.2702    2.76    0.00626 **
service       2.5097     0.4495    5.58 0.00000008 ***
```


The Overall F Test

- There is one more hypothesis test that R (and all other stat packages) do for you automatically.
- It is called the "Overall F test"
- It tests the null hypothesis:

$$H_0: \beta_1 = \beta_2 = \cdots = \beta_k = 0$$

i.e. there is no relation between X an Y

You always want to reject his

- What is the alternative hypothesis?

at least 1 Bi ≠ 0

Motivating the Test

- Under the null hypothesis, there are no X variables in the model.
- If there are no X variables in the model, then SSR=0 and SST=SSR+SSE=SSE.
- However, if at least one X variable is useful, then SSR does not equal 0, and ideally, if some X variables are useful, SSR>SSE
- So we compare SSR to SSE in some fashion.

The Test Statistics

- The test statistics for testing

$$H_0: \beta_1 = \beta_2 = \cdots = \beta_k = 0$$

- Is given by

$$f = \frac{(SSR)/k}{SSE/(n-k-1)}$$

- We reject for large values of *f*.

The *F* distribution

- How large is large?
- The *F distribution* tell us what kind of values we expect to get for f, when the null is true.
- In particular, under the null,

$$f \sim F_{k,n-k-1}$$

- Which is the F distribution with k numerator degrees of freedom and n-k-1 denominator degrees of freedom.

The Decision Rule

$$\text{reject } H_0 \text{ if } f \geq f_{k,n-k-1,\alpha}$$

- One can calculate the p-value by hand using the R `pf()` command.
- For our purposes, we will simply read the p-value from the regression output. (Score!).

Example

$$H_o : \beta_1 = \beta_2 = \beta_3 = 0$$
$$vs.$$
$$H_a : At\ least\ one\ \beta_i \neq 0$$

Conclusion?

```
Call:
lm(formula = price ~ size + age + lotsize)


Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -16.0580    19.0710  -0.842 0.417712
size          4.1462     0.7512   5.520 0.000181 ***
age          -0.2361     0.8812  -0.268 0.793730
lotsize       4.8309     0.9011   5.361 0.000230 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 6.894 on 11 degrees of freedom
Multiple R-squared:  0.9161,    Adjusted R-squared:  0.8932
F-statistic: 40.03 on 3 and 11 DF,  p-value: 3.278e-06
```


Example

$$H_o : \beta_1 = \beta_2 = \beta_3 = 0$$
$$vs.$$
$$H_a : At\ least\ one\ \beta_i \neq 0$$

Conclusion?

```
> fit=lm(piq~brain+height+weight)
> summary(fit)

Call:
lm(formula = piq ~ brain + height + weight)

Residuals:
   Min     1Q Median     3Q    Max
-32.73 -12.09  -3.84  14.17  51.69

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept) 111.378186  62.971483    1.77  0.08591 .
brain         2.060200   0.563455    3.66  0.00086 ***
height       -2.732402   1.229522   -2.22  0.03302 *
weight        0.000716   0.197064    0.00  0.99712
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 20 on 34 degrees of freedom
Multiple R-squared:  0.295,     Adjusted R-squared:  0.233
F-statistic: 4.74 on 3 and 34 DF,  p-value: 0.00722
```


F's and t's

- If you just want to test whether one coefficient is 0 it might appear that we now have two ways. The t test and the F test.
- It turns out they are equivalent.
- Mathematically, $f=t^2$

Example

$$\boxed{3.871^2 = 14.2}$$

```
> fit=lm(price~food)
> summary(fit)

Call:
lm(formula = price ~ food)

Residuals:
   Min     1Q Median     3Q    Max
-23.49  -8.31  -1.85   7.11  42.59

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   -3.871     10.047   -0.39  0.70046
food           1.640      0.436    3.76  0.00022 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 12 on 193 degrees of freedom
Multiple R-squared:  0.0683,    Adjusted R-squared:  0.0635
F-statistic: 14.2 on 1 and 193 DF,  p-value: 0.000223
```


Variable selection

- If we have k variables, and assuming a constant term in each model, there are 2^k-1 possible subsets of variables, not counting the null model with no variables. How do we select a subset for our model?
- Two main approaches: Stepwise regressions and all possible regressions.
- A point to note-modelling is hard.

Stepwise Regression

- A full regression course is required to fully understand modeling, but it will be beneficial to begin the thought process of how to work with a lot of variables.
- One easy way to do this is to perform something called “backward stepwise regression”.
- Under this scheme, you start with all the variables in the model, and remove them one by one.

Variable Selection: Backward Stepwise Regression

The way hypothesis testing works, you are only allowed to remove *one variable at a time* from the model.

So one way we build models as follows:

•Start with all variables in the model

•at each step, delete the least important variable from the remaining ones based on largest p-value above 0.05.

•stop when you can’t delete any more.

Example: Football data; what variables contribute to a winning season ?

Column	Count	Name
C1	28	wins
C2	28	rush
C3	28	pass
C4	28	patt
C5	28	pcomp
C6	28	pint
C7	28	penalty
C8	28	fumble
C9	28	rushopp
C10	28	passopp
C11	28	pattopp
C12	28	pcompopp
C13	28	piopp

The Full Model

```
> fit=lm(wins~rush+pass+patt+pcomp+pint+penalty+fumbles+rushopp+passopp+pattopp+pcompopp+piopp)
> summary(fit)

Call:
lm(formula = wins ~ rush + pass + patt + pcomp + pint + penalty +
    fumbles + rushopp + passopp + pattopp + pcompopp + piopp)

Residuals:
     Min       1Q   Median       3Q      Max
-1.88194 -0.96564  0.09151  0.75299  2.61849

Coefficients:
               Estimate  Std. Error t value Pr(>|t|)
(Intercept) -2.05583056  9.90592844  -0.208  0.83838
rush         0.00152605  0.00143588   1.063  0.30469
pass         0.00289591  0.00142320   2.035  0.05994 .
patt        -0.01161437  0.01950896  -0.595  0.56049
pcomp        0.00911988  0.02916689   0.313  0.75883
pint        -0.06647342  0.10589890  -0.628  0.53964
penalty     -0.00097561  0.00466386  -0.209  0.83712
fumbles     -0.01763890  0.09579048  -0.184  0.85637
rushopp      0.00004805  0.00177364   0.027  0.97874
passopp     -0.00590162  0.00151141  -3.905  0.00141 **
pattopp      0.06102059  0.02325585   2.624  0.01917 *
pcompopp    -0.02233248  0.01909735  -1.169  0.26049
piopp       -0.07731961  0.11164524  -0.693  0.49918
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 1.578 on 15 degrees of freedom
Multiple R-squared:  0.8738,    Adjusted R-squared:  0.7729
F-statistic: 8.659 on 12 and 15 DF,  p-value: 0.0001019
```


least valuable variable : rushopp
l t l is closest to 0
remember we want l t l > 1.96
also p-value is furthest away from 0.05

Remove RUSHOPP (why?)

```
> fit1=lm(wins~rush+pass+patt+pcomp+pint+penalty+fumbles+passopp+pattopp+pcompopp+piopp)
> summary(fit1)

Call:
lm(formula = wins ~ rush + pass + patt + pcomp + pint + penalty +
    fumbles + passopp + pattopp + pcompopp + piopp)

Residuals:
     Min       1Q   Median       3Q      Max
-1.85945 -0.97045  0.09558  0.74859  2.62532

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept) -1.9319164  8.5078391  -0.227 0.823241
rush         0.0015089  0.0012475   1.209 0.244040
pass         0.0028924  0.0013725   2.107 0.051205 .
patt        -0.0114803  0.0182716  -0.628 0.538666
pcomp        0.0089597  0.0276548   0.324 0.750148
pint        -0.0672078  0.0991226  -0.678 0.507442
penalty     -0.0009653  0.0045009  -0.214 0.832886
fumbles     -0.0176719  0.0927435  -0.191 0.851278
passopp     -0.0058881  0.0013816  -4.262 0.000596 ***
pattopp      0.0608653  0.0218231   2.789 0.013135 *
pcompopp    -0.0222535  0.0182747  -1.218 0.240984
piopp       -0.0773400  0.1081002  -0.715 0.484643
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 1.528 on 16 degrees of freedom
Multiple R-squared:  0.8738,    Adjusted R-squared:  0.7871
F-statistic: 10.07 on 11 and 16 DF,  p-value: 0.0000305
```

We should see:
R2 adjusted going up
Se going down

Remove the variable with the highest p-value above .05 then refit

Easier way to do this removal

- There is a model update command in R

```
> fit=lm(wins~rush+pass+patt+pcomp+pint+penalty+fumbles+rushopp+passopp+pattopp+pcompopp+piopp)
> fit1=update(fit,.~.-rushopp)
> summary(fit1)

Call:
lm(formula = wins ~ rush + pass + patt + pcomp + pint + penalty +
    fumbles + passopp + pattopp + pcompopp + piopp)

Residuals:
     Min       1Q   Median       3Q      Max
-1.85945 -0.97045  0.09558  0.74859  2.62532

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept) -1.9319164  8.5078391  -0.227 0.823241
rush         0.0015089  0.0012475   1.209 0.244040
pass         0.0028924  0.0013725   2.107 0.051205 .
patt        -0.0114803  0.0182716  -0.628 0.538666
pcomp        0.0089597  0.0276548   0.324 0.750148
pint        -0.0672078  0.0991226  -0.678 0.507442
penalty     -0.0009653  0.0045009  -0.214 0.832886
fumbles     -0.0176719  0.0927435  -0.191 0.851278
passopp     -0.0058881  0.0013816  -4.262 0.000596 ***
pattopp      0.0608653  0.0218231   2.789 0.013135 *
pcompopp    -0.0222535  0.0182747  -1.218 0.240984
piopp       -0.0773400  0.1081002  -0.715 0.484643
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 1.528 on 16 degrees of freedom
Multiple R-squared:  0.8738,    Adjusted R-squared:  0.7871
F-statistic: 10.07 on 11 and 16 DF,  p-value: 0.0000305
```

Now remove FUMBLES

```
> fit2=update(fit1,.~.-fumbles)
> summary(fit2)

Call:
lm(formula = wins ~ rush + pass + patt + pcomp + pint + penalty +
    passopp + pattopp + pcompopp + piopp)

Residuals:
    Min      1Q  Median      3Q     Max
-1.9524 -0.9999  0.1174  0.7394  2.5911

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) -1.491490   7.952383  -0.188 0.853448
rush         0.001499   0.001211   1.239 0.232350
pass         0.002954   0.001296   2.279 0.035874 *
patt        -0.011864   0.017638  -0.673 0.510242
pcomp        0.008403   0.026709   0.315 0.756890
pint        -0.064290   0.095117  -0.676 0.508189
penalty     -0.001362   0.003876  -0.351 0.729567
passopp     -0.005902   0.001340  -4.405 0.000387 ***
pattopp      0.060776   0.021191   2.868 0.010660 *
pcompopp    -0.023211   0.017065  -1.360 0.191542
piopp       -0.072794   0.102403  -0.711 0.486809
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.484 on 17 degrees of freedom
Multiple R-squared:  0.8736,    Adjusted R-squared:  0.7992
F-statistic: 11.74 on 10 and 17 DF,  p-value: 0.000008598
```


After a while get to this

```
> fit10=lm(wins~rush+pass+pint+passopp+pattopp)
> summary(fit10)

Call:
lm(formula = wins ~ rush + pass + pint + passopp + pattopp)

Residuals:
    Min      1Q  Median      3Q     Max
-2.0626 -1.0763  0.0480  0.6624  3.2261

Coefficients:
              Estimate Std. Error t value  Pr(>|t|)
(Intercept) -5.7428336  5.2888387  -1.086   0.28930
rush         0.0020463  0.0010463   1.956   0.06330 .
pass         0.0029797  0.0005326   5.595 0.0000126 ***
pint        -0.1106437  0.0620384  -1.783   0.08831 .
passopp     -0.0053287  0.0009882  -5.392 0.0000205 ***
pattopp      0.0401539  0.0120817   3.324   0.00308 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.393 on 22 degrees of freedom
Multiple R-squared:  0.8558,    Adjusted R-squared:  0.823
F-statistic: 26.11 on 5 and 22 DF,  p-value: 0.00000001461
```


Drop PINT

```
> fit10=lm(wins~rush+pass+passopp+pattopp)
> summary(fit10)

Call:
lm(formula = wins ~ rush + pass + passopp + pattopp)

Residuals:
    Min      1Q  Median      3Q     Max
-2.6100 -1.1772  0.2459  0.8287  2.3167

Coefficients:
               Estimate  Std. Error t value  Pr(>|t|)
(Intercept) -10.2492232   4.8615000  -2.108   0.04611 *
rush          0.0025855   0.0010481   2.467   0.02150 *
pass          0.0029576   0.0005571   5.309 0.0000217 ***
passopp      -0.0055535   0.0010255  -5.415 0.0000168 ***
pattopp       0.0448382   0.0123391   3.634   0.00139 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.457 on 23 degrees of freedom
Multiple R-squared:  0.8349,    Adjusted R-squared:  0.8062
F-statistic: 29.09 on 4 and 23 DF,  p-value: 0.00000001067
```

Why do we stop here ? all | t | are above 1.96
all p-values < 0.05

How does this compare to previous model?

Wait, Se and R2 adj were actually better in the prev model
The 1.96 and 0.05 have been assumed as hard and fast rules
Conclusion: We dropped a variable that is marginally needed.
We should be more careful at the very end.

R can (sort of) do it automatically

- There are better methods than this but someone wrote a function called model.select to do this.
- Load the function into R as follows

```
source("http://people.fas.harvard.edu/~mparzen/stat100/model_select.txt")
```


Running model.select()

```
> model.select(fit,verbose=FALSE)

Call:
lm(formula = wins ~ rush + pass + passopp + pattopp)

Coefficients:
(Intercept)         rush         pass      passopp      pattopp
  -10.24922      0.00259      0.00296     -0.00555      0.04484

 model.select(fit,verbose=TRUE)
-------------STEP  9 -------------
 The drop statistics :
Single term deletions

Model:
wins ~ rush + pass + passopp + pattopp
        Df Sum of Sq   RSS  AIC F value    Pr(>F)
<none>                48.9 25.6
rush     1      12.9  61.8 30.2    6.09  0.0215 *
pass     1      59.9 108.7 46.0   28.19 0.000022 ***
passopp  1      62.3 111.1 46.6   29.32 0.000017 ***
pattopp  1      28.0  76.9 36.3   13.20  0.0014 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Call:
lm(formula = wins ~ rush + pass + passopp + pattopp)

Coefficients:
(Intercept)         rush         pass      passopp      pattopp
  -10.24922      0.00259      0.00296     -0.00555      0.04484
```


The built in method in R

```
step(fit,model="backward")

Step:  AIC=23.81
wins ~ rush + pass + pint + passopp + pattopp

          Df Sum of Sq     RSS    AIC
<none>                  42.685 23.806
- pint     1     6.171  48.856 25.587
- rush     1     7.422  50.106 26.294
- pattopp  1    21.431  64.116 33.198
- passopp  1    56.419  99.104 45.391
- pass     1    60.736 103.421 46.585

Call:
lm(formula = wins ~ rush + pass + pint + passopp + pattopp)

Coefficients:
(Intercept)        rush        pass        pint     passopp     pattopp
  -5.742834    0.002046    0.002980   -0.110644   -0.005329    0.040154
```

The `step` function in R minimizes AIC – details in more advanced courses.

All Subsets Regression

- This procedure runs all 1 variable models, all 2 variable models, all 3 variable models and so on.
- The idea is to pick the model that has the adjusted R-2 [or some other measure].
- The output looks cool at least.

All subsets regression

- The function is call `regsubsets` and is in the `leaps` package:

```
library(leaps)

fit=regsubsets(wins~rush+pass+patt+pcomp+pint+penalty+fumb
les+rushopp+passopp+pattopp+pcompopp+piopp,data=mydata)

plot(fit,scale="adjr2")
```


The Output