

Brief Introduction to Stata

AAE636

I. Introduction

Why we use Stata?

1. It is easy to use with an easy point-and-click interface and an intuitive command syntax;
2. It is fast and accurate;
3. It is powerful;
4. It runs on different platforms;
5. It is widely used and it is available.
6. It extends into more complex applications using its programming features.

Stata SE 10 is installed in the Halvorson-Ebling computer lab. First log in the department computer using your aae user name and password (students from other departments can use guest account to get access to the aae computers), and then you can find it at

Start\All Programs\Statistics\StataSe 10

A very good resource for you to get help when you learn and use Stata is at:

<http://www.ats.ucla.edu/stat/stata/>, there are many lectures, examples and datasets for beginners and those working on specific projects.

The books we tend to follow are:

Statistics with Stata: Updated for version 9 by Lawrence C. Hamilton, 2006

Stata Manuals, Release 10. Focus will be on three manuals: Data Management, Graphics, and Reference

II. Starting a Session

The first step may be to change the directory of where STATA is looking to save and read information.

Suppose you have a flashdrive in the computer labeled E: and suppose further that it also has a directory for this class and a subdirectory for stasession. To direct stata to this subdirectory, use the 'cd' command.

```
. cd e:\aae636\stasession
```

It would be typical to begin a session to create a log file that you can refer back to later. This is critical if you need to recall steps you took in modifying data, generating new data, or creating really groovy and hip figures and graphs.

After bringing up STATA, type in the command line: `. log using [filename]` or `. log using [filename.log]`

Alternatively, you can use the FILE tab and click on 'log' to "begin" a session.

Note: if you did not change the directory, pay attention to where the file is being created.

When you are done with STATA, close the log file from the FILE tab or type “log close” in the command line.

III. Data Entry

1. Input a dataset created from the Excel spreadsheet with the suffix **csv**

```
. insheet using (directory)\filename.csv
```

Example:

```
. insheet using L:\public_html\Notes\data\USDA_Beefdata.csv
```

Alternatively, change the directory first and bring the data in without the directory information.

Example:

```
. cd L:\public_html\Notes\data\
```

```
.insheet using USDA_Beefdata.csv
```

2. Input a Stata dataset with the suffix **dta**. This is default, thus, we only need the dataname or dataname.dta.

Note: Sometimes, the data file may be too big to be read in. So before we read a large dataset, we will have to reset the amount of memory allocated to Stata.

For example, if we want to read a dataset called collegedistance.dta

```
. cd z:stata\
```

```
. use collegedistance      \* open the dataset collegedistance.dta *\
```

There appears the error message in red: **no room to add more observations....**

Given that you actually want all the data you are trying to load, the solution is to increase the amount of memory allocated to the data area using the set memory command;

We increase memory by doing the following:

```
. set memory 10m
```

Now, there is no error message and the dataset is successfully read in.

IV. Data Management

1. Managing the variables in the dataset

(i) Rename variables

```
. replace a = b           \* rename a variable called 'a' to one called 'b' *\
```

(ii) Generate new variables

```
. gen loga=log(a) \*generate a new variable called loga by taking the log of variable a*\
```

```
. gen c =a*b           \* generate a new variable called c by taking the products of a and b *\
```

```
. egen d =mean(a) \* generate a new variable called d with the mean value of variable a *\
```

(iii) Recode a variable to a new one

```
. gen a_new=a           \* generate a new variable called a_new from variable a *\
```

```
. replace a_new=1 if (a < 10)           \* a_new=1 if a is less than 10 *\
```

```
. replace a_new=2 if (a >= 10) & (a < 20) \* a_new=2 if a is between 10 and 20 *\
```

```
. replace a_new=3 if (a >=20)           \* a_new=3 if a is greater than 20 *\
```

Note, a_new is now a dummy variable with values 1,2,or 3 depending on the values of the variable a.

2. Managing missing values

Missing values are shown as dots for numeric variables and blanks for string variables.

```
. mvdecode old mv(-999) \* convert a value coded as -999 in another data set to a Stata missing value *\
```

```
. mvencode new mv(-999) \* convert Stata missing value to new values *\
```

3. Dropping variables

```
. drop a           \* drop variable a from the dataset *\
```

```
. drop in 1/5       \* drop the first five observations from the dataset *\
```

```
. drop if x= .       \*get rid of missing values in the variable x*\
```

```
. keep a           \* keep variable a while dropping the others *\
```

```
. keep in -5/1       \* keeps the last 5 observations in the dataset *\
```

4. Sorting variables

```
. sort scores       \* sorts the dataset using variable 'scores' from small to large numbers *\
```

. sort scores in 1/7 * does the same only for the first 7 observations in the dataset *\

. sort scores, stable * same as sort, but leaves identical 'scores' in original order*\

. sort x y * arranges variable x in an ascending order and the observations having identical x are arranged such that y is in an ascending order *\

. gsort –scores * sorts in descending order, see sort and gsort for other differences *\

. gsort x -y * sorts x in ascending order, and for ties in x sorts y in descending order *\

5. Append

. use *filename1* *open the disk dataset or “using” dataset *\

. use *filename2* *open the master dataset *\

. append using *filename1* *append the filename1 to the filename2 *\

6. Merge

One-to-one (merges the corresponding observations of different datasets)

. use *filename1* *open the disk dataset or “using” dataset *\

. use *filename2* *open the master dataset *\

. merge using *filename1* *join the two datasets using a one-to-one merge*\

Match-merge (merges the observations with matching values of the specified variables in different datasets)

. use *filename1* *open the disk dataset or “using” dataset *\

. use *filename2* *open the master dataset *\

merge *varname* using *filename*, sort

7. Joinby (within groups, forms all pair wise combinations)

. use *filename1* *open the disk dataset or “using” dataset *\

. use *filename2* *open the master dataset *\

. joinby *varname* using *filename*, sort *join, within groups formed by varname, observations in the different datasets *\

8. Assert

. assert gender= =”male” | gender= =”female” *assert a binary variable contains only male or female*\

9. .codebook and .inspect * displays useful information about the data*\

Example, from our beefsurvey, lets look at our age variable:

```
.codebook qnd6
```

```
-----  
qnd6 (unlabeled)  
-----
```

```
type: numeric (byte)  
range: [1,11] units: 1  
unique values: 11 missing .: 0/8002  
mean: 7.70245  
std. dev: 2.18894  
percentiles: 10% 25% 50% 75% 90%  
              5 6 8 10 10
```

Note: It is always a good idea to run these types of diagnostics on your data when you first load them. Any issue or problem you can find and fix in the early stages saves tons of time and gives you confidence to proceed.

10. Collapse (make dataset of summary statistics)

```
. collapse price mpg, by(foreign) \*return mean price mean mpg for the different values of foreign*\
```

```
. collapse (75) price, by(foreign) \*return 75th percentile of price for the different values of foreign*\
```

```
. collapse (mean) price (median) mpg, by(foreign) \*return mean price and median mpg for the different values of foreign*\
```

11. Count (counts observations satisfying specified conditions)

```
. count if income < 0 \*check if income is nonnegative*\
```

```
. count if gender == 3 \*make sure that the binary variable “gender” contains two values only*\
```

12. Describe Provides a line by line description of each data column.

Example:

```
. describe caseid qns1 qns2 qnd6
```

```
storage display value  
variable name type format label variable label
```

```
caseid    int    %8.0g
qns1     byte  %8.0g
qns2     byte  %8.0g
qnd6     byte  %8.0g
```

13. Functions: Students should read section in STATA manual on all the available functions and distributions at their disposal.

14. Infile: Earlier, we showed how to use the `.insheet` command. `insheet` is really only one from a list of ways to bring in non-Stata (stata files are `.dta` format) formats. See also `xmlsave` for interfacing with Excel.

15. Label *manipulates label*\

```
. label variable x "miles per gallon"
```

16. `. list` * display the content of a variable in memory dataset *\

Example: list the 10th-14th observations for question `qnd6`.

```
. list qnd6 in 10/14
```

```
+-----+
| qnd6 |
|-----|
10. |  8 |
11. | 10 |
12. |  8 |
13. |  3 |
14. |  5 |
+-----+
```

17. `tabulate y` * yield a frequencies analysis and percentage table of variable `y` *\

Example: `tabulate qn4`

```
. tabulate qn4
```

qn4	Freq.	Percent	Cum.
1	455	8.96	8.96
2	455	8.96	17.92
3	250	4.92	22.85
4	2,199	43.31	66.16
5	1,718	33.84	100.00

V. Graphics

Stem-and-leaf plot

```
. stem x          \* Stem-and-leaf plot for variable x *\
```

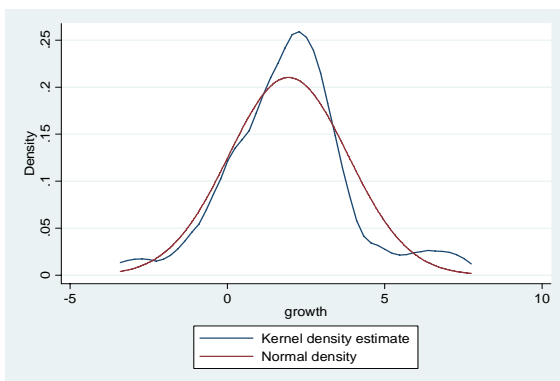
Search on “stem” for additional options for stem and leaf plots.

Kernal density plot

```
. kdensity x, normal \* draw kernel density function of x with a comparison of normal distribution*\
```

For example,

```
. kdensity growth, normal
```



We see a kernel density estimate of the growth rate of GDP and we can compare it with the red normal distribution function.

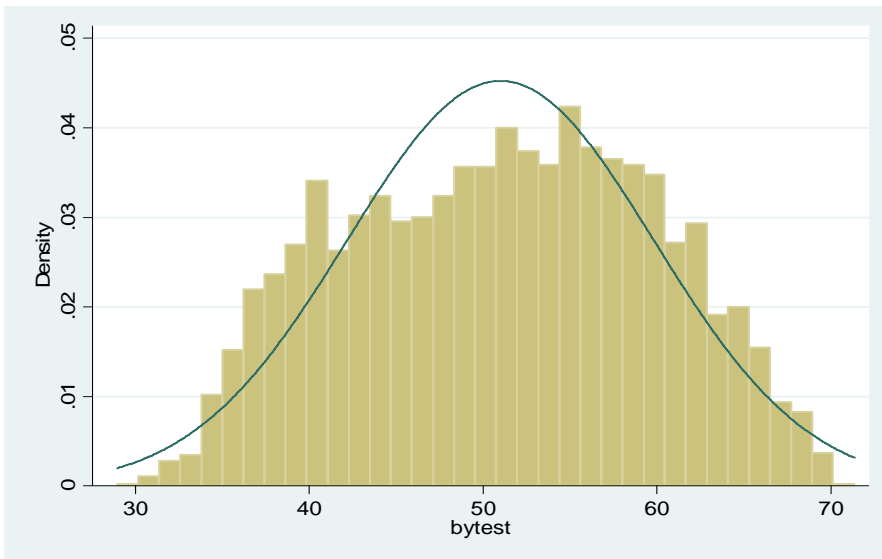
Histogram

```
. hist x, normal \* draw histogram of variable x and a normal distribution line *\
```

```
. hist x, by(a) \* draw histogram of variable x by categories in dummy variable a *\
```

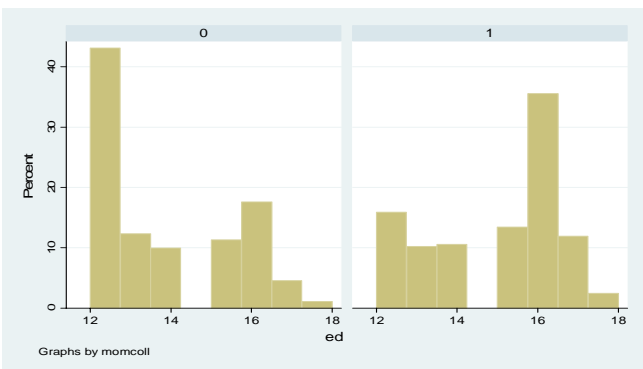
For example,

```
. hist bytest, normal
```



From this graph, we can see the histogram of test scores and compare it with the normal distribution.

```
. hist ed, by(momcoll) percent bin(8)
```



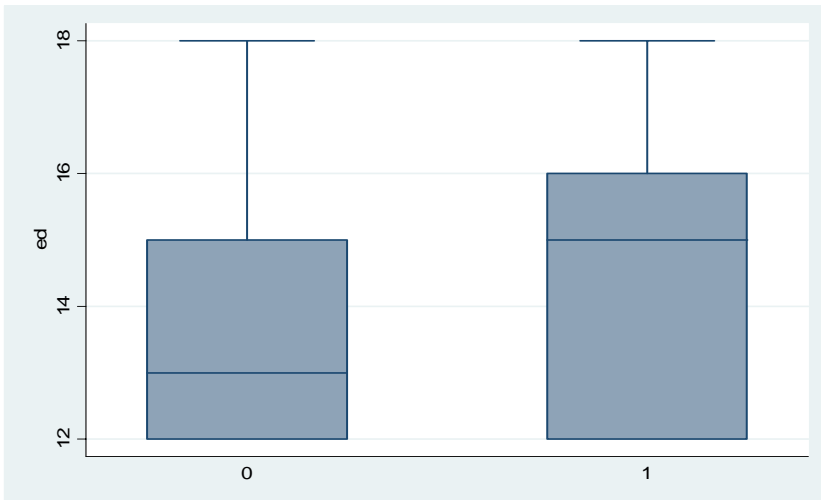
From this graph, we can see the histogram of education years of those people whose mothers don't go to college (momcoll = 0) and of those whose mothers go to college (momcoll = 1).

Box plot

```
. graph box x, over(a) \* draw box plots of variable x by separate groups in a *\
```

For example,

```
. graph box ed, over(incomehi)
```



From this graph, we can see the box plots of education years of those people who come from a low income family (incomehi = 0) and of those who come from a high income family (incomehi = 1). Obviously, the average years of education is higher for those people from high income families.

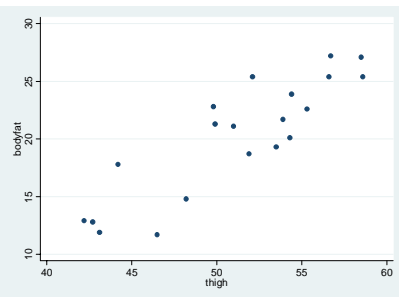
Scatter plot

```
. scatter x y, by(a) \* draw scatter plots of variables x and y by separate groups in a *\
```

```
. graph matrix x y z, half \* draw multiple scatter plots of variables x, y and z *\
```

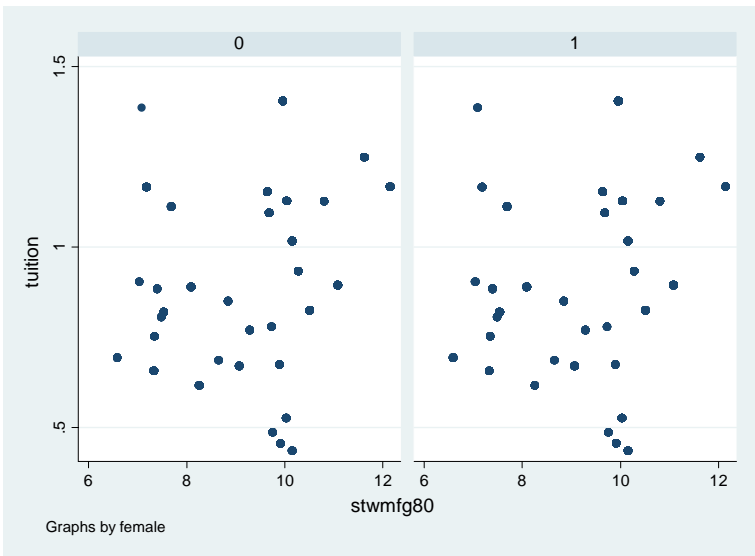
For example,

```
. scatter bodyfat thigh
```



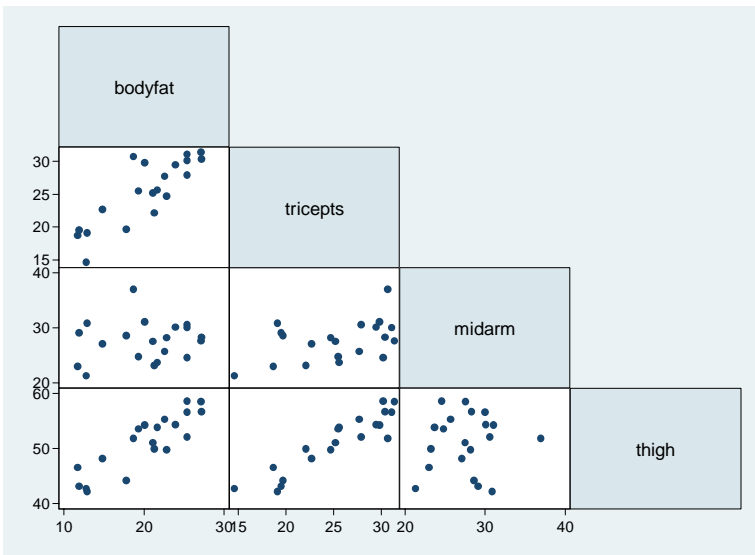
From this scatter plot graph, we can see a clear linear relationship between the volume of bodyfat and the size of thigh.

```
. scatter tuition stwmfg80, by(female)
```



From this graph, we see the scatter plots of tuition against stwmfg80 for both male and female groups.

```
. graph matrix bodyfat triceps midarm thigh, half
```



This graph shows the scatter plots of every two variables of the four variables. The option half lets us only see the half of the whole picture which is perfectly fine because the other half is symmetric and exactly the same.

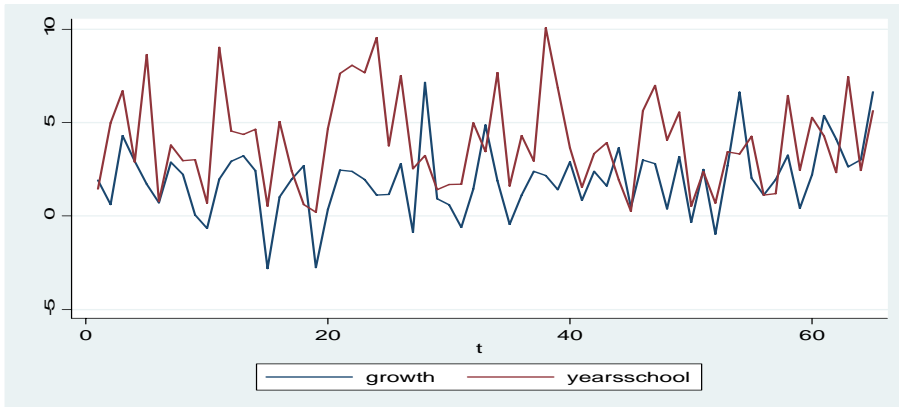
Line plot

```
. line x y z          \* draw lines of variables x and y against variable z *\
```

```
. graph twoway connected x z \* create connected-line plots of variable x against variable z *\
```

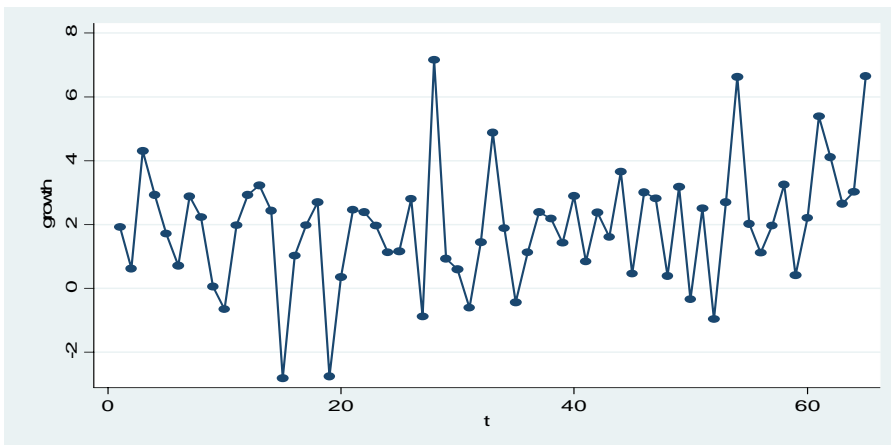
For example,

```
. line growth yearsschool t
```



t is the country id. This graph shows the patterns of the growth rate of countries' GDP and the average years of schooling of the adult residents in those countries.

```
. graph twoway connected growth t
```



This graph shows the actual dots and the connected line of the growth rate of countries' GDP.

VI. Statistic Analysis

```
. sum a b c \* yield basic descriptive statistics of variables a, b and c *\
```

For example,

```
. sum growth rgdp60 tradeshare
```

Variable	Obs	Mean	Std. Dev.	Min	Max
growth	65	1.942715	1.89712	-2.811944	7.156855
rgdp60	65	3103.785	2512.657	366.9999	9895.004
tradeshare	65	.564703	.2892703	.140502	1.992616

. tabstat a b c, stats(mean sd sdmean n) * yield the mean, standard deviation, standard error of the mean and the number of observations of variables a, b and c *\

For example,

. tabstat growth rgdp60 tradeshare, stats(mean sd sdmean n)

stats | growth rgdp60 trades~e

	growth	rgdp60	tradeshare
mean	1.942715	3103.785	.564703
sd	1.89712	2512.657	.2892703
se(mean)	.2353087	311.6567	.0358796
N	65	65	65

. tabstat X, stat(n mean sd p25 p50 p75) by(a) * yield the number of observations, mean, standard deviation, 25th,50th and 75th percentile of variable X by categories of a *\

For example,

. tabstat y, stat(n mean sd p25 p50 p75) by(b)

Summary for variables: y

by categories of: b (B)

b	N	mean	sd	p25	p50	p75
1	8	2.75	1.488048	2	2.5	3
2	8	3.5	.9258201	3	3.5	4
3	8	6.25	1.035098	5.5	6	7
4	8	9	1.309307	8	9	10
Total	32	5.375	2.756225	3	5	7.5

. tabulate a, summarize(X) * yield the mean, standard deviation and frequencies of variable X by categories of a *\

For example,

. tabulate b, summarize(y)

Summary of Y			
B	Mean	Std. Dev.	Freq.
1	2.75	1.4880476	8
2	3.5	.9258201	8
3	6.25	1.0350983	8
4	9	1.3093073	8
Total	5.375	2.7562246	32

Often times, we are interested in finding the correlations among the variables. We use corrltion or corr command to calculate them.

```
. corr a b c          \* yield the correlations among the variables a, b and c *\
```

For example,

```
. corr triceps bodyfat midarm
```

```
(obs=20)
```

```
      | triceps bodyfat  midarm
```

```
-----+-----
```

```
triceps | 1.0000
```

```
bodyfat | 0.8433 1.0000
```

```
midarm  | 0.4578 0.1424 1.0000
```

t-tests

paired t-test

```
. ttest a=b          \* test whether the means of variables a and b are the same *\
```

For example,

From the USDA survey, two interesting questions have the same scale:

qn4. Overall, do you approve or disapprove of the Beef Checkoff program?

1=strongly disapprove, 2=somewhat disapprove, 3=neither, 4=somewhat approve, 5= strongly approve, 6,7 DK, NA

and

qn18: If a referendum were held tomorrow on whether or not to continue the \$1-per-head checkoff program, how likely would you be to vote to continue the program?

1=very unlikely, 2=somewhat unlikely, 3=neither, 4=somewhat likely, 5= very likely, 6,7 DK, NA

```
. ttest qn4=qn18
```

Paired t test

```
-----
```

Variable	Obs	Mean	Std. Err.	Std. Dev.	[95% Conf. Interval]
----------	-----	------	-----------	-----------	----------------------

```
-----+-----
qn4 | 5077 3.841048 .0173468 1.236011 3.807041 3.875055
qn18 | 5077 4.057317 .0195049 1.389786 4.019079 4.095555
-----+-----
```

```
diff | 5077 -.2162695 .0132003 .9405639 -.2421478 -.1903911
-----+-----
```

```
mean(diff) = mean(qn4 - qn18)          t = -16.3836
Ho: mean(diff) = 0                    degrees of freedom = 5076
Ha: mean(diff) < 0      Ha: mean(diff) != 0      Ha: mean(diff) > 0
Pr(T < t) = 0.0000      Pr(|T| > |t|) = 0.0000      Pr(T > t) = 1.0000
```

Note : We had already removed the responses equal to 6 or 7 from the survey. From the above table, we can see that the mean of approval/disapproval is 3.84 and the mean of continue/discontinue is 4.05. The significance test of three null hypotheses that the means are the same is calculated and reported. At the 5% level of significance, this null hypothesis that $\text{mean}(qn4 - qn18) > 0$ is not rejected. The hypothesis that $\text{mean}(qn4 - qn18) = 0$ is rejected and the hypothesis that $\text{mean}(qn4 - qn18) < 0$ is also rejected. Notice that the 95% confidence interval in the diff | row does not contain zero.

one sample t-test

```
. ttest a=x          \* test whether the mean of variable a is x *\
```

For example,

```
. ttest qn4=3
```

One-sample t test

```
-----+-----
Variable |  Obs   Mean  Std. Err.  Std. Dev.  [95% Conf. Interval]
-----+-----
qn4 |  5077  3.841048  .0173468  1.236011  3.807041  3.875055
-----+-----
```

```
mean = mean(qn4)          t = 48.4844
Ho: mean = 3              degrees of freedom = 5076
Ha: mean < 3      Ha: mean != 3      Ha: mean > 3
Pr(T < t) = 1.0000      Pr(|T| > |t|) = 0.0000      Pr(T > t) = 0.0000
```

Note : from the above table, we can see that the mean of approve/disapprove is 3.84. The test that the mean is equal to 3 is soundly rejected. The one-sided tests give obvious results.

two sample t-test

. ttest x, by(a) * test the difference between the averages of variable x using separate groups in variable a which is a dummy *\

For example, using the approve/disapprove question (qn4) and testing if mean answers by gender are different,

. ttest qn4, by(gendum1)

Two-sample t test with equal variances

```

-----
Group |  Obs   Mean  Std. Err.  Std. Dev.  [95% Conf. Interval]
-----+-----
    0 |  359  3.674095  .064752   1.226874   3.546753   3.801437
    1 |  4718  3.853752  .0179932  1.23591   3.818477   3.889027
-----+-----
combined |  5077  3.841048  .0173468  1.236011   3.807041   3.875055
-----+-----
diff |      -1.1796569  .0676302          -3.122413  -.0470725
-----
diff = mean(0) - mean(1)                t = -2.6565
Ho: diff = 0                            degrees of freedom = 5075
Ha: diff < 0                            Ha: diff != 0                Ha: diff > 0
Pr(T < t) = 0.0040                      Pr(|T| > |t|) = 0.0079        Pr(T > t) = 0.9960

```

Note : from the above table, we can see that the mean of test score of male is 3.85 (gendum1=1) and the mean of response score of female is 3.67. The significance test of three null hypotheses that the means are the same is calculated and reported. At the 5% level of significance, this null hypothesis that mean(0)-mean(1)> 0 is not rejected while the other two are null hypotheses. Notice that the 95% confidence interval in the diff | row does not contain zero.

Mann-Whitney U test.

The Mann-Whitney U test (also called the Mann-Whitney-Wilcoxon (MWW), Wilcoxon rank-sum test, or Wilcoxon-Mann-Whitney test) is a nonparametric test for assessing whether two samples of observations come from the same distribution.

The null is that the two samples are drawn from a single population, and therefore that their probability distributions are equal. It requires the two samples to be independent, and the observations to be ordinal or continuous measurements, i.e. one can at least say, of any two observations, which is the greater. It is commonly thought that the MWW test tests for differences in medians but this is not strictly true. The U-test is an extension of the Wilcoxon (1945) test that was developed for equal sample sizes. The MWW test considers data with different sample sizes Mann and Whitney (1947).

The U formula can be calculated:

Arrange all the observations into a single ranked series. That is, rank all the observations without regard to which sample they are in.

Add up the ranks for the observations which came from the smaller sample: call this U. Compute $U' = n_1(n_1 + n_2 + 1) - U$. The smaller of the two values of U and U' are used to determine significance.

Assuming this is U, the mean and standard deviation are: $\mu_U = \frac{n_1(n_1 + n_2 + 1)}{2}$, $\sigma_U = \sqrt{\frac{n_1 n_2 (n_1 + n_2 + 1)}{12}}$. The

Z-stat is calculated $Z = (U - \mu_U) / \sigma_U$. In STATA, the procedure is “ranksum” using a dummy variable to signify a change in the sample.

Example: test if responses about approval of the beef checkoff program are from different probability distributions based on how informed you are.

```
. ranksum qnd4, by (bin_inform)
```

Two-sample Wilcoxon rank-sum (Mann-Whitney) test

```
bin_inform |  obs  rank sum  expected
-----+-----
      0 | 1558  4045746  3955762
      1 | 3519  8844757  8934741
-----+-----
combined | 5077 12890503 12890503
unadjusted variance 2.320e+09
adjustment for ties -8.316e+08
```

adjusted variance 1.488e+09

Ho: qnd4(bin_in~m==0) = qnd4(bin_in~m==1)

$z = 2.332$

Prob > |z| = 0.0197

The null hypothesis that the informed respondents are from the same distribution as those “not informed” is rejected.

Matrix Operations

All matrix commands begin with the word **matrix**. We can either create matrices by inputting element by element using **matrix** or **matrix define**, or we can create matrices from the variables of an existing data set by joining the columns using **mkmat**.

```
. matrix A=(3,5,4\6,9,10\7,3,8)          \* create a 3X3 matrix called A *\
```

Note: to input a matrix, we enclose the element in the parentheses with commas to separate elements and backslashes to separate rows.

```
. matrix list A                          \* display matrix A *\
```

```
A[3,3]
      c1  c2  c3
r1    3   5   4
r2    6   9  10
r3    7   3   8
```

```
. matrix B=(3\6\9)                       \* create a 3X1 column vector called B *\
```

```
. matrix list B                          \* display matrix B *\
```

```
B[3,1]
      c1
r1    3
r2    6
r3    9
```

```
. matrix C=(3,5,4)                       \* create a 1X3 row vector called C *\
```

```
. matrix list C                          \* display matrix C *\
```

```
C[1,3]
      c1  c2  c3
r1    3   5   4
```

```
. mkmat impressions adexpenditures, matrix(D) \* create a matrix from the existing
dataset and call this new matrix D *\
```

```
. matrix list D                          \* display matrix D *\
```

```
D[21,2]
```

	impressions	adexpenditures
r1	32.099998	50.099998
r2	99.599998	74.099998
r3	11.7	19.299999
r4	21.9	22.9
r5	60.799999	82.400002
r6	78.599998	40.099998
r7	92.400002	185.89999
r8	50.700001	26.9
r9	21.4	20.4
r10	40.099998	166.2
r11	40.799999	27
r12	10.4	45.599998
r13	88.900002	154.89999
r14	12	5
r15	29.200001	49.700001
r16	38	26.9
r17	10	5.6999998
r18	12.3	7.5999999
r19	23.4	9.1999998
r20	71.099998	32.400002
r21	4.4000001	6.0999999

```
. matrix E=A,B \* join matrices A and B columns by row *\
```

```
. matrix list E \* display matrix E *\
```

```
E[3,4]
  c1 c2 c3 c1
r1  3  5  4  3
r2  6  9 10  6
r3  7  3  8  7
```

```
. matrix F=A\C \* join matrices A and C rows by column *\
```

```
. matrix list F \* display matrix F *\
```

```
F[4,3]
  c1 c2 c3
r1  3  5  4
r2  6  9 10
r3  7  3  8
r1  3  5  4
```

```
. matrix U=J(3,1,1) \* create a 3X1 matrix U with all 1's *\
```

```
. matrix list U \* display matrix U *\
```

```
U[3,1]
  c1
r1  1
r2  1
r3  1
```

```
. matrix U=J(3,3,10) \* create a 3X3 matrix U with all 10's *\
```

```
. matrix list U \* display matrix U *\
```

```

symmetric U[3,3]
  c1 c2 c3
r1 10
r2 10 10
r3 10 10 10

. matrix I=I(3)                                \* create a 3X3 identity matrix I *\

. matrix list I                                \* display matrix I *\

symmetric I[3,3]
  c1 c2 c3
r1 1
r2 0 1
r3 0 0 1

. matrix S=diag(B)                             \* create a diagonal matrix from matrix B *\

. matrix list S                                \* display matrix S *\

symmetric S[3,3]
  r1 r2 r3
r1 3
r2 0 6
r3 0 0 7

. matrix S=diag(vecdiag(A))                   \* create a diagonal matrix from matrix A *\

. matrix list S                                \* display matrix S *\

symmetric S[3,3]
  c1 c2 c3
c1 3
c2 0 9
c3 0 0 8

. scalar r=rowsof(D)                          \* give r the number of how many rows are in matrix D *\

. display r                                    \* display r or the number of rows in matrix D *\
21

. scalar c=colsof(D)                          \* give c the number of how many columns are in matrix D *\

. display c                                    \* display r or the number of rows in matrix D *\
2

. matrix G=3*A                                \* create matrix G by multiplying matrix A with 3 *\

. matrix list G                                \* display matrix G *\

G[3,3]
  c1 c2 c3
r1 9 15 12
r2 18 27 30
r3 21 9 24

. matrix H=A+G                                \* create matrix H by addition of matrix A and G *\

```

```

. matrix list H                                     \* display matrix H *\

H[3,3]
  c1  c2  c3
r1  12  20  16
r2  24  36  40
r3  28  12  32

. matrix I=G-A                                     \* create matrix I by subtraction of matrix G and A *\

. matrix list I                                     \* display matrix H *\

I[3,3]
  c1  c2  c3
r1   6  10  8
r2  12  18  20
r3  14   6  16

. matrix J=A*G                                     \* create matrix J by multiplication of matrix A and G *\

. matrix list J                                     \* display matrix J *\

J[3,3]
  c1  c2  c3
r1  201  216  282
r2  426  423  582
r3  285  258  366

. matrix AT=A'                                     \* create matrix AT by transposing matrix A *\

. matrix list AT                                     \* display matrix AT *\

AT[3,3]
  r1  r2  r3
c1   3   6   7
c2   5   9   3
c3   4  10   8

. matrix INVA=inv(A)                               \* create matrix INVA by inverting matrix A *\

. matrix list INVA                                 \* display matrix INVA *\

INVA[3,3]
      r1          r2          r3
c1          .75          -.5          .25
c2   .39285714  -.07142857  -.10714286
c3  -.80357143   .46428571  -.05357143

. scalar d=det(A)                                  \* calculate the determinant of matrix A and give the value to d *\

. display d                                         \* display d or the determinant of matrix A *\
56

. matrix U=J(rowsof(A),1,1)                         \* create a 3X1 matrix U with all 1's *\

. matrix S=U'*A                                     \* create matrix S by multiplying matrix A with the transpose
of matrix U *\

```

```

. matrix list S

S[1,3]
      c1  c2  c3
c1  16  17  22

. matrix cm=S/rowsof(A)          \* create matrix cm by dividing matrix S with the number
of rows in matrix A *\

. matrix list cm

cm[1,3]
      c1      c2      c3
c1  5.3333333  5.6666667  7.3333333

. matrix D=diag(B)              \* create a diagonal matrix D from matrix B *\

. matrix S=cholesky(D) \* return the Cholesky decomposition of symmetric and positive
definite matrix D *\

. matrix list S

symmetric S[3,3]
      r1      r2      r3
r1  1.7320508
r2      0  2.4494897
r3      0      0  2.6457513

The following is an example of calculating correlation coefficient of vectors X and Y
using matrix operations we learned above. Check and see if you understand.

. matrix X=(9\6\5\7\8\6\4)

. matrix Y=(4\3\7\9\6\1\3)

. matrix U=J(rowsof(X),1,1)

. matrix X=X-U*((U'*X)/rowsof(X))

. matrix Y=Y-U*((U'*X)/rowsof(Y))

. matrix Y=Y-U*((U'*Y)/rowsof(Y))

. matrix Y=(4\3\7\9\6\1\3)

. matrix Y=Y-U*((U'*Y)/rowsof(Y))

. matrix c=(X'*Y)*syminv(cholesky(X'*X)*cholesky(Y'*Y))

. matrix list c

symmetric c[1,1]
      c1
c1  .20647119

. matrix A=(32)                \* create a 1X1 matrix A with number 32 *\

. matrix list A

```

```

symmetric A[1,1]
      c1
r1 32

. display A                                \* can't use display because A is a matrix *\
type mismatch
r(109);

. scalar z=A[1,1]                          \* give the number in matrix A to a scalar z *\

. display z                                \* display z or the only element in matrix A *\
32

```

Note: command **display** cannot be used to show matrices.

OLS Estimates

```

. cd z:\stata                                \* enter the working directory *\
. use collegedistance                        \* open the dataset called collegedistance *\
. regress ed bytest tuition

```

Source	SS	df	MS			
Model	2837.22332	2	1418.61166	Number of obs =	3796	
Residual	9650.15919	3793	2.54420226	F(2, 3793) =	557.59	
Total	12487.3825	3795	3.29048287	Prob > F =	0.0000	
				R-squared =	0.2272	
				Adj R-squared =	0.2268	
				Root MSE =	1.5951	

	ed	Coef.	Std. Err.	t	P> t	[95% Conf. Interval]	
	bytest	.0988043	.0029802	33.15	0.000	.0929614	.1046472
	tuition	-.1626279	.0926838	-1.75	0.079	-.3443428	.019087
	_cons	8.938586	.1628357	54.89	0.000	8.619332	9.25784

The estimation results are all shown in the above table with the coefficients, the standard errors of the coefficients, the t statistics, the P value and the 95% confidence interval of the true coefficients. All we need to do is to type in the command **regress**, and the dependent variable and then the predictors. We might be interested in how we get those results. Next is an example of doing the regression by ourselves by manually typing in the formula.

```

. matrix define I=I(3796)                    \* create a 3796X3796 identity matrix I *\
matsize too small
      You have attempted to create a matrix with more than 400 rows or columns or
      to fit a model with more than 400 variables plus ancillary parameters.  You
      need to increase matsize using the set matsize command; see help matsize.
r(908);

```

Oops, we need to increase the matsize because the default is too small and our matrix is pretty big with more than 400 rows and columns.

```

. set matsize 4000                            \* increase the matrix size *\

```

Current memory allocation

settable	current value	description	memory usage (1M = 1024k)
set maxvar	5000	max. variables allowed	1.733M
set memory	10M	max. data space	10.000M
set matsize	4000	max. RHS vars in models	122.406M

			134.139M

```
. matrix define I=I(3796)                \* create a 3796X3796 identity matrix I *\

. matrix define con=(vecdiag(I))'        \* build the constant vector which is a 3796X1
matrix of ones *\

. mkmat bytest tuition,matrix(x1)        \* create a 3796X2 matrix x1 with the two independent
variables bytest and tuition *\

. matrix define x=con,x1                 \* create matrix x by combining the constant vector with
matrix x1 *\

. matrix beta_c=inv(x'*x)                \* create matrix beta_c by inverting the multiplicaiton of the
transpose of matrix x and matrix x *\

. matrix list beta_c
symmetric beta_c[3,3]
      r1      bytest      tuition
r1      .01042192
bytest  -.00016101  3.491e-06
tuition -.00213182  -.00001865  .00337642

. mkmat ed, matrix(y)                   \* create a 3796X1 matrix y by including the dependent variable *\

. matrix beta_b=x'*y                    \* create matrix beta_b by multiplying matrix y with the transpose
of matrix x *\

. matrix beta=beta_c*beta_b              \* create matrix beta by multiplying matrix beta_b with the
matrix beta_c *\

. matrix list beta                       \* display matrix beta which is the OLS coefficients *\

beta[3,1]
      ed
r1      8.9385856
bytest  .0988043
tuition -.16262786

. regress ed bytest tuition
```

Source	SS	df	MS	Number of obs =	3796
Model	2837.22332	2	1418.61166	F(2, 3793) =	557.59
				Prob > F	= 0.0000

Residual		9650.15919	3793	2.54420226		R-squared	=	0.2272

Total		12487.3825	3795	3.29048287		Adj R-squared	=	0.2268

		Coef.	Std. Err.	t	P> t	[95% Conf. Interval]		

bytest		.0988043	.0029802	33.15	0.000	.0929614	.1046472	
tuition		-.1626279	.0926838	-1.75	0.079	-.3443428	.019087	
_cons		8.938586	.1628357	54.89	0.000	8.619332	9.25784	

Compare the beta we calculated using the formula with the results produced by regress command, they are the same. The F-test is testing whether the coefficients of bytest and tuition are both equal to 0. We can get the same F-test by typing in **test bytest tuition.**

```
. matrix define dev=vecdiag((beta_c))' \* create 3X1 vector dev by taking the diagonal
element in matrix beta_c *\

. matrix define dev1=2.54420226*dev \* create 3X1 vector dev1 by multiplying matrix dev
with the variance of the error terms *\

. matrix list dev1

dev1[3,1]
           r1
      r1  .02651546
  bytest 8.882e-06
  tuition .00859029

. display 0.0029802^2
8.882e-06
```

The numbers in vector dev1 are the variances of the three coefficients. Next is an example of the relation between F-test and t-test.

```
. regress ed tuition
```

Source		SS	df	MS		Number of obs =	3796	

Model		40.7251389	1	40.7251389		F(1, 3794) =	12.41	
Residual		12446.6574	3794	3.28061607		Prob > F =	0.0004	

Total		12487.3825	3795	3.29048287		R-squared =	0.0033	

		Coef.	Std. Err.	t	P> t	[95% Conf. Interval]		

tuition		.3653029	.1036811	3.52	0.000	.1620268	.5685789	
_cons		13.49572	.0991345	136.14	0.000	13.30136	13.69008	

From the above example with only one predictor besides the constant, we can see the t-test statistic for tuition is the square root of the F-test statistic ($12.41=3.52^2$).

Check for heteroskedasticity

One way we check for heteroskedasticity of the error term is to draw the twoway scatter plots of the residuals against either the one of the predictors or the fitted value of the dependent variable. If there is an obvious trend in the plots, we should be cautious of the existence of heteroskedasticity.

```
. regress ed tuition bytest \* run an OLS regression of education years on the level
of tuition and the test scores *\
```

Source	SS	df	MS			
Model	2837.22332	2	1418.61166	Number of obs =	3796	
Residual	9650.15919	3793	2.54420226	F(2, 3793) =	557.59	
Total	12487.3825	3795	3.29048287	Prob > F =	0.0000	
				R-squared =	0.2272	
				Adj R-squared =	0.2268	
				Root MSE =	1.5951	

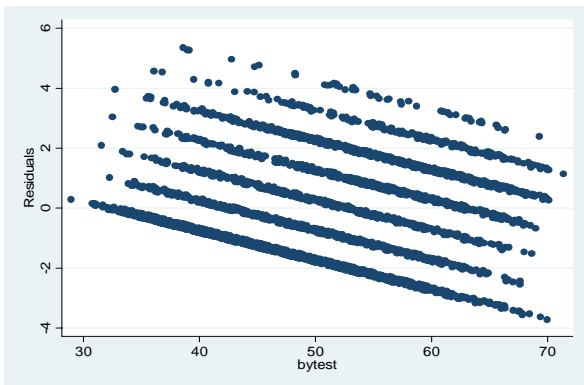
	ed	Coef.	Std. Err.	t	P> t	[95% Conf. Interval]	
tuition		-.1626279	.0926838	-1.75	0.079	-.3443428	.019087
bytest		.0988043	.0029802	33.15	0.000	.0929614	.1046472
_cons		8.938586	.1628357	54.89	0.000	8.619332	9.25784

```
. predict p1 \* generate the fitted values of the previous model and put them in a new
variable called p1 *\
```

```
. predict r1, residual \* generate the residuals of the model and put them in a new
variable called r1 *\
```

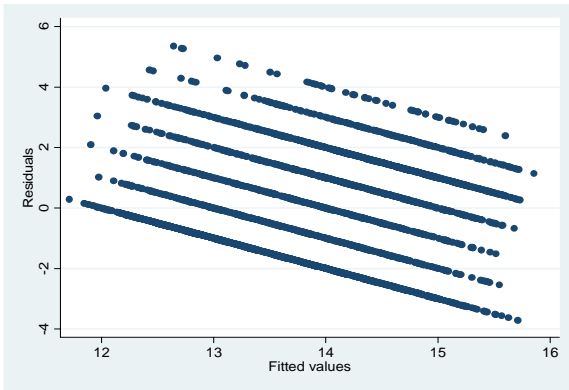
```
. twoway scatter r1 bytest \* create the scatter plot of residuals against one
of the predictors bytest *\
```

or . rvpplot bytest



```
. twoway scatter r1 p1 \* create the scatter plot of residuals against the
fitted values of the model*\
```

or . rvfplot



From the above two residual plots, we can clearly see that there is a downward sloping trend. We should be skeptical about heteroskedasticity.

The other way is to test for heteroskedasticity by calculating the test statistic, such as Breusch-Pagan test.

```
. estat hettest,rhs
```

```
Breusch-Pagan / Cook-Weisberg test for heteroskedasticity
```

```
Ho: Constant variance
```

```
Variables: bytest tuition
```

```
chi2(2) = 69.45
```

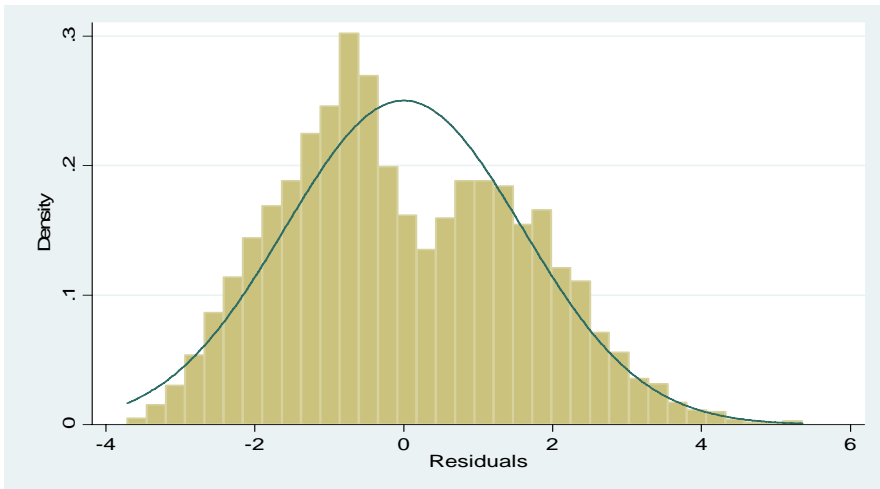
```
Prob > chi2 = 0.0000
```

From the above Breusch-Pagan test, we find that the null hypothesis of constant variance is rejected.

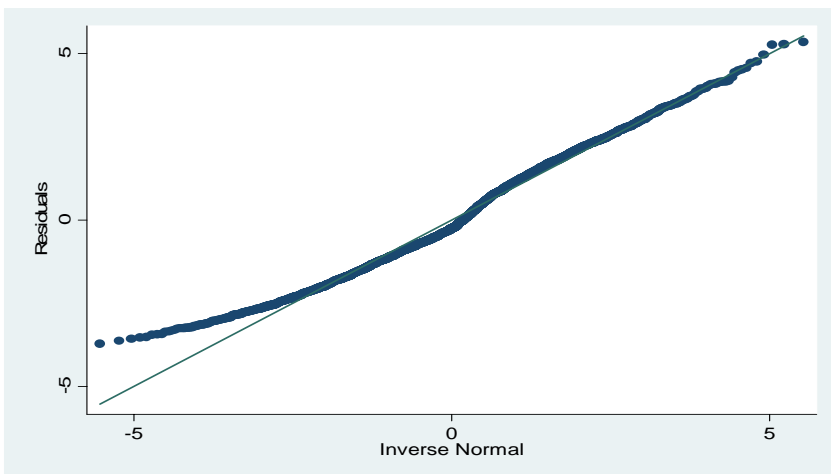
Check for Normality

For the same model in the last section, we can also use graphs to check for the assumption of normality of the residuals. One is using the histogram graph to see if the distribution of the residuals is normal, the other is using the q-q norm to see if the points are on the linear line.

```
. hist r1,normal \* create a histogram of the residuals *\
(bin=35, start=-3.7127204, width=.25910809)
```



```
. qnorm r1                                \* create a q-q norm plot of the residuals *\
```



Monte-Carlo Simulation

Monte-Carlo simulation is a tool for understanding the properties of a model or statistic under certain conditions. In real life, we almost never know the true values of the parameters, so by using Monte-Carlo simulation, we can perform the same type of analysis on data with known characteristics. Let's get started with Monte-Carlo simulation with a pretty easy example.

```
. clear                                    \* removes data from stata's memory *\
. set obs 100                               \* changes the number of observations in the current dataset *\
obs was 0, now 100

. scalar a=2                                \* let a = 2 *\
. scalar b=3                                \* let b = 3 *\
. scalar wt=10                              \* let wt = 2 *\
. gen x=uniform()*100                       \* generate vector x ~Uniform(0,100) *\
```

```
. gen e=invnorm(uniform())*wt          \* generate vector e ~norm(0,100) *\
. gen y=a+b*x+e                        \* generate vector y = a+bx+e *\
. regress y x                          \* regress y on x *\
```

Source	SS	df	MS	Number of obs =	100
Model	647370.611	1	647370.611	F(1, 98) =	5735.50
Residual	11061.3409	98	112.870825	Prob > F =	0.0000
				R-squared =	0.9832
				Adj R-squared =	0.9830
Total	658431.952	99	6650.8278	Root MSE =	10.624

	y	Coef.	Std. Err.	t	P> t	[95% Conf. Interval]
x		2.987067	.039442	75.73	0.000	2.908795 3.065338
_cons		3.279918	2.154869	1.52	0.131	-.9963498 7.556186

From the above table, we find that the estimated a is 3.279918, the estimated b is 2.987067 and the variance of residuals is 112.870825. And the true a is 2, true b is 3 and the true variance of the error terms is 100. The estimate of a is far from the true value and the other two are ok.

How can we improve our estimates? We can repeat the procedures many times and calculate the average of the estimates which will give us results closer to the true values. The following program is used to generate the random variables, do the regression and return the estimates that we are interested in. It is very convenient to use program in this case because we need to repeat the process multiple times.

```
. program regr, rclass
1. drop _all
2. set obs 100
3. scalar a=2
4. scalar b=3
5. scalar wt=10
6. gen x=uniform()*100
7. gen e=invnorm(uniform())*wt
8. gen y=a+b*x+e
9. regress y x
10. return scalar b1=_b[x]
11. end

. simulate "regr" b1=r(b1), reps(100)          \* repeat the program by 100 times *\

command:      regr
statistic:    b1          = r(b1)

. sum          \* create the summary statistics for b through those 100 repetitions *\
```

Variable	Obs	Mean	Std. Dev.	Min	Max
b1	100	2.999383	.0319701	2.932761	3.100489

Now, it is very obvious that the average of the generated 100 b estimates is 2.999383 which is closer to the true value of b than before.

Units don't matter

The units of variables will not affect the estimation results. Here is an example.

```
. cd z:\                \* enter the working directory *\
. use growth            \* open the data set growth.dta *\
. regress growth rgdp60 tradeshare \* do the OLS regression *\
```

Source	SS	df	MS			
Model	31.695228	2	15.847614	Number of obs =	65	
Residual	198.644829	62	3.20394886	F(2, 62) =	4.95	
Total	230.340057	64	3.59906339	Prob > F =	0.0102	
				R-squared =	0.1376	
				Adj R-squared =	0.1098	
				Root MSE =	1.79	

	Coef.	Std. Err.	t	P> t	[95% Conf. Interval]	
growth						
rgdp60	.0000902	.0000901	1.00	0.321	-.00009	.0002703
tradeshare	2.426423	.7827234	3.10	0.003	.8617814	3.991065
_cons	.2927	.6006422	0.49	0.628	-.9079666	1.493367

Now, we want to change the unit of variable rgdp60 by multiplying the original data with 100 and let's see if the change would affect the regression results.

```
. gen rgdp60_revised=100* rgdp60 \* generate a new variable by multiplying the old one
with 100 *\
. regress growth rgdp60_revised tradeshare \* do the OLS regression using the new
variable *\
```

Source	SS	df	MS			
Model	31.695228	2	15.847614	Number of obs =	65	
Residual	198.644829	62	3.20394886	F(2, 62) =	4.95	
Total	230.340057	64	3.59906339	Prob > F =	0.0102	
				R-squared =	0.1376	
				Adj R-squared =	0.1098	
				Root MSE =	1.79	

	Coef.	Std. Err.	t	P> t	[95% Conf. Interval]	
growth						
rgdp60_rev~d	9.02e-07	9.01e-07	1.00	0.321	-9.00e-07	2.70e-06
tradeshare	2.426423	.7827234	3.10	0.003	.8617814	3.991065
_cons	.2927	.6006422	0.49	0.628	-.9079666	1.493367

From the above two graphs, we can see that the coefficient of the changed variable is 1/100 of the original coefficient, and the same thing happened to the standard error, thus the t-statistic stays the same. Estimation results of other variables are the same before and after the change.

Identifying Influential Points and Robust Regression

We should be cautious about unusual data points in linear models since they can affect the results of the analysis, and their presence might be a signal that the model fails to capture important characteristics of the data.

Outliers are extreme observations and it is a common practice to distinguish between two types of outliers. Outliers in the response variable are called residual outliers and outliers with respect to the independent variables are called leverage points. Leverage points' response variables need not be outliers, but they may almost uniquely determine regression coefficients. They may also cause the standard errors of regression coefficients to be much smaller than they would be if the observation were excluded.

Let's run the following linear model and detect the residual outliers.

```
. regress growth rgdp60 tradeshare yearsschool rev_coups assassinations \* regression *\
```

Source	SS	df	MS	Number of obs = 65		
Model	82.6634812	5	16.5326962	F(5, 59)	=	6.61
Residual	147.676576	59	2.50299281	Prob > F	=	0.0001
-----				R-squared	=	0.3589
-----				Adj R-squared	=	0.3045
Total	230.340057	64	3.59906339	Root MSE	=	1.5821

growth	Coef.	Std. Err.	t	P> t	[95% Conf. Interval]	
rgdp60	-.0004693	.0001482	-3.17	0.002	-.0007659	-.0001728
tradeshare	1.561696	.7579475	2.06	0.044	.0450462	3.078345
yearsschool	.5748461	.1393379	4.13	0.000	.2960316	.8536606
rev_coups	-2.157503	1.110292	-1.94	0.057	-4.379191	.0641853
assassinati~s	.3540784	.4773943	0.74	0.461	-.6011854	1.309342
_cons	.4897603	.6895996	0.71	0.480	-.8901254	1.869646

```
. predict sres, rstandard \* generate the standardized residuals and put them in a new variable called sres *\
```

```
. gen index=_n \* generate the index number for each observation *\
```

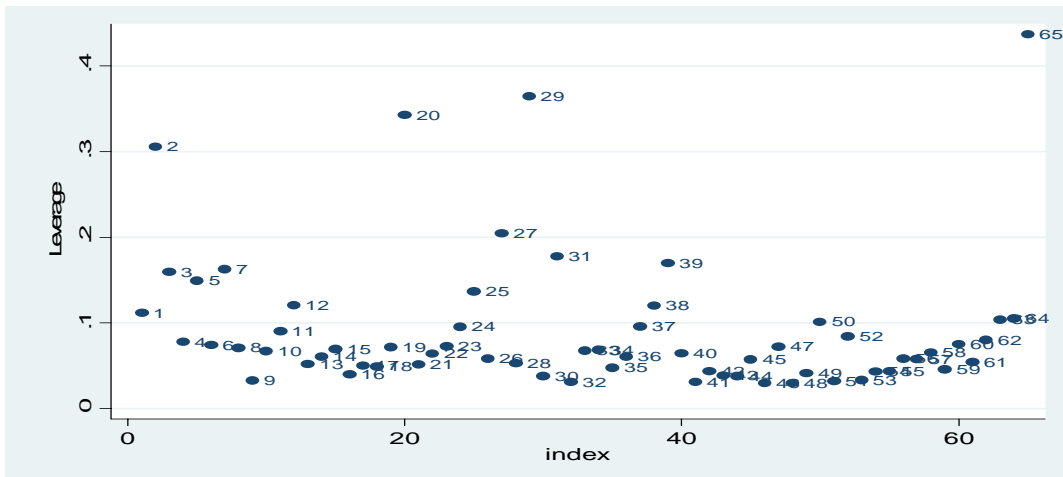
```
. twoway scatter sres index, mlabel(index) \* create the scatter plot of residuals with the observation number labeled *\
```



From the residual plot, we find that observation #28 is an outlier because it has the largest standardized residual. Next, let's find the leverage points.

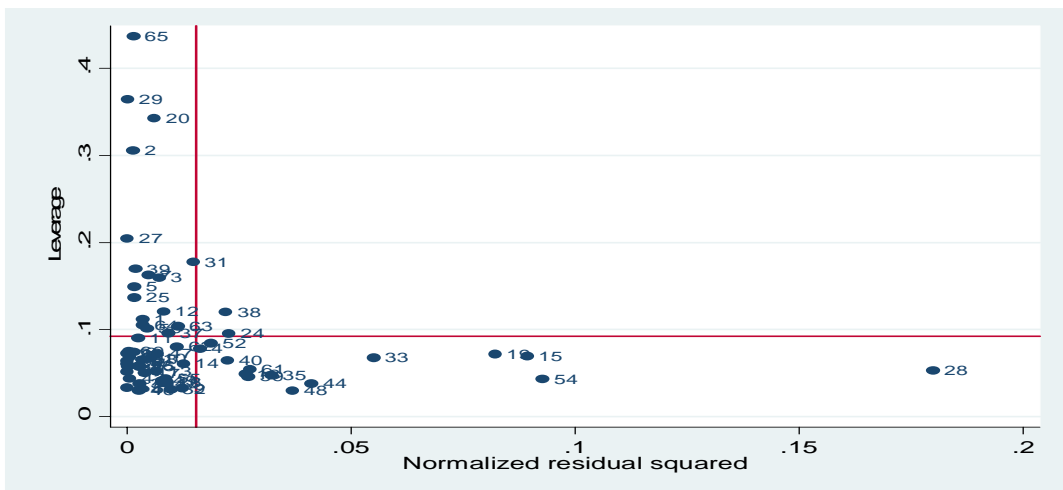
```
. predict lev,lev          \* generate the leverages for each observation and put
them in the new variable called lev *\

. twoway scatter lev  index, mlabel(index)    \* create the leverage plot of each
observation *\
```



From the leverage plot, we find that observation #65 has a high leverage and it is a leverage point. We can also plot the standardized residuals and the leverages on the same graph using the following command.

```
. lvr2plot,mlabel(index) \* create the leverage and residual plot for each observation *\
```



As before, observation #65 is a leverage point and observation #28 is a residual outlier. What to do with outliers? Only when there is a direct evidence that they represent an error in recoding, a miscalculation, a malfunctioning of equipment, or some similar type of circumstance, we can discard these outliers. Sometimes, detection of outliers itself might be interesting.

But not all outliers are influential points. To find influential points, we need to calculate cook's distance. The threshold level is $4/(n-k)$, where n is the number of observations and k is the number of parameters. So, when the cook's distance value is greater than the threshold level for a certain observation, this observation is an influential point.

```
. predict d1,cooksdist \* calculate cook's distance for each observation and put them in
a new variable called d1 *\
```

```
. list if d1>4/65 \* list those observations with cook's distance greater than the
threshold level, or list all the influential points *\
```

```
+-----+
15. |          country_name |      growth | oil |      rgdp60 | trades~e | yearss~l | rev_co~s | assassi~s |
|          Zaire | -2.811944 | 0 | 488.9999 | .3523176 |          .54 | .1481481 | .0555556 |
+-----+
|          sres      |          index |          lev      |          d1      |
| -2.38042          |          15      |          .0694673 |          .0705025 |
+-----+
```

```
+-----+
19. |          country_name |      growth | oil |      rgdp60 | trades~e | yearss~l | rev_co~s | assassi~s |
|          Niger | -2.751478 | 0 | 531.9999 | .4258372 |          .2 | .1333333 |          0 |
+-----+
|          sres      |          index |          lev      |          d1      |
| -2.285589          |          19      |          .0717397 |          .0672875 |
+-----+
```

```
+-----+
28. |          country_name |      growth | oil |      rgdp60 | trades~e | yearss~l | rev_co~s | assassi~s |
| Korea, Republic of | 7.156855 | 0 | 904.0001 | .4889496 |          3.23 | .3333333 |          .1 |
+-----+
|          sres      |          index |          lev      |          d1      |
| 3.347654          |          28      |          .0527102 |          .1039303 |
+-----+
```

In this data set, we have 3 influential points (#15, #19 and #28). So #65 is not an influential point according to its cook's distance value. Next, let's see the difference of a robust regression with the regular OLS regression in term of dealing with influential points.

```
. rreg growth rgdp60 tradeshare yearsschool rev_coups assassinations,gen(weight)
\* do a robust regression of the original model and generate a new variable called weight
with the values given by Stata *\
```

```
Huber iteration 1: maximum difference in weights = .65476598
Huber iteration 2: maximum difference in weights = .12914625
Huber iteration 3: maximum difference in weights = .04162505
Biweight iteration 4: maximum difference in weights = .2887742
Biweight iteration 5: maximum difference in weights = .0328026
Biweight iteration 6: maximum difference in weights = .01435069
Biweight iteration 7: maximum difference in weights = .0041822
```

```
Robust regression                               Number of obs =          65
                                                F( 5,          59) =          7.04
                                                Prob > F           =          0.0000
```

```
+-----+
growth |          Coef.      Std. Err.      t      P>|t|      [95% Conf. Interval]
+-----+
rgdp60 | -.0003611          .0001328      -2.72  0.009      -.0006268      -.0000954
```

tradeshare		1.522225	.678972	2.24	0.029	.1636055	2.880845
yearsschool		.4570549	.1248194	3.66	0.001	.2072919	.7068178
rev_coups		-2.750625	.994603	-2.77	0.008	-4.740821	-.7604292
assasinati~s		.5411897	.4276515	1.27	0.211	-.3145389	1.396918
_cons		.5993589	.6177457	0.97	0.336	-.6367474	1.835465

It is obvious that the robust regression results are different from the regular OLS regression. There are iterations and weights. What are those weights?

```
. sort weight      \* sort the new variable weight in an increasing order *\
. list index  country_name weight sres d1 in 1/10  \* list the first 10 observations in
the order of their weights *\
```

	index	country_name	weight	sres	d1
1.	28	Korea, Republic of	.00004558	3.347654	.1039303
2.	54	Taiwan, China	.25993772	2.390591	.0427185
3.	15	Zaire	.32268373	-2.38042	.0705025
4.	19	Niger	.35223043	-2.285589	.0672875
5.	33	Thailand	.41845569	1.865102	.0421595
6.	44	Portugal	.62347313	1.588249	.0165539
7.	61	Cyprus	.69703952	1.307616	.0164026
8.	48	South Africa	.71260868	-1.497822	.011409
9.	35	Senegal	.71348047	-1.411489	.0165892
10.	59	Jamaica	.75582161	-1.292636	.013352

Observation #28 is an influential point and it is given the least weight in the robust regression. Observation #54 is given the second least weight and so on so forth. Hence, in the robust regression, Stata gives different data points different weights. The more unusual the data point is the less weight is given to it. The less the weight it has, the less the effect it has on the whole model. But in OLS regression, every data point is given the same weight 1, so unusual data points are not taken into account.

Note: the weight is not given according to the size of cook's distance.

Joint Hypothesis Test (F-test)

```
. regress growth rgdp60 tradeshare      \* regress the restricted model *\
```

Source	SS	df	MS	Number of obs = 65		
Model	31.695228	2	15.847614	F(2, 62) =	4.95	
Residual	198.644829	62	3.20394886	Prob > F =	0.0102	
				R-squared =	0.1376	
				Adj R-squared =	0.1098	
Total	230.340057	64	3.59906339	Root MSE =	1.79	

growth	Coef.	Std. Err.	t	P> t	[95% Conf. Interval]	
rgdp60	.0000902	.0000901	1.00	0.321	-.00009	.0002703

tradeshare		2.426423	.7827234	3.10	0.003	.8617814	3.991065
_cons		.2927	.6006422	0.49	0.628	-.9079666	1.493367

```
. regress growth rgdp60 tradeshare yearsschool rev_coups assassinations \* regress the full model *\
```

Source	SS	df	MS	Number of obs = 65		
Model	82.6634812	5	16.5326962	F(5, 59)	=	6.61
Residual	147.676576	59	2.50299281	Prob > F	=	0.0001
				R-squared	=	0.3589
				Adj R-squared	=	0.3045
Total	230.340057	64	3.59906339	Root MSE	=	1.5821

growth	Coef.	Std. Err.	t	P> t	[95% Conf. Interval]	
rgdp60	-.0004693	.0001482	-3.17	0.002	-.0007659	-.0001728
tradeshare	1.561696	.7579475	2.06	0.044	.0450462	3.078345
yearsschool	.5748461	.1393379	4.13	0.000	.2960316	.8536606
rev_coups	-2.157503	1.110292	-1.94	0.057	-4.379191	.0641853
assasinati~s	.3540784	.4773943	0.74	0.461	-.6011854	1.309342
_cons	.4897603	.6895996	0.71	0.480	-.8901254	1.869646

```
. matrix R=(0,0,0,1,-1,0\0,0,0,0,1,-1) \* construct the R matrix *\
. matrix r=(0\0) \* construct the r vector *\
. gen con=1 \* construct the constant vector *\
. mkmat con rgdp60 tradeshare yearsschool rev_coups assassinations,matrix(x) \* construct the x matrix by combining the constant vector with the independent variables *\
. mkmat growth, matrix(y) \* construct the y vector of dependent variable *\
. matrix beta=inv(x'*x)*x'*y \* construct the beta by using the OLS formula *\
. matrix list beta \* display the full model coefficients *\
beta[6,1]
      growth
      con   .48976027
      rgdp60 -.00046935
      tradeshare 1.5616957
      yearsschool .57484608
      rev_coups -2.1575029
      assassinations .35407841
. matrix F_n=(R*beta-r)'*inv(R*inv(x'*x)*R')*(R*beta-r)/2
. matrix F_d=147/(65-6)
. matrix F=F_n*inv(F_d) \* calculate the F statistic in equation (6.12) in the lecture notes *\
```

Or we can simply use the following command to do the job:

```
. test yearsschool rev_coups assassinations \* test whether the coefficients are jointly
equal to 0 *\

( 1) yearsschool = 0
( 2) rev_coups = 0
( 3) assassinations = 0

      F( 3, 59) = 6.79
      Prob > F = 0.0005
```

We reject the null hypothesis at the 5% significance level. So at least one of the three coefficients is different from 0.

Autocorrelation

```
. cd z:\                                \* enter the working directory *\
z:\

. use JEC.dta                            \* open the data set called JEC *\

. gen t=_n                                \* generate a new index variable t *\

. tsset t                                 \* tell Stata that this is a time series data set *\
   time variable: t, 1 to 328

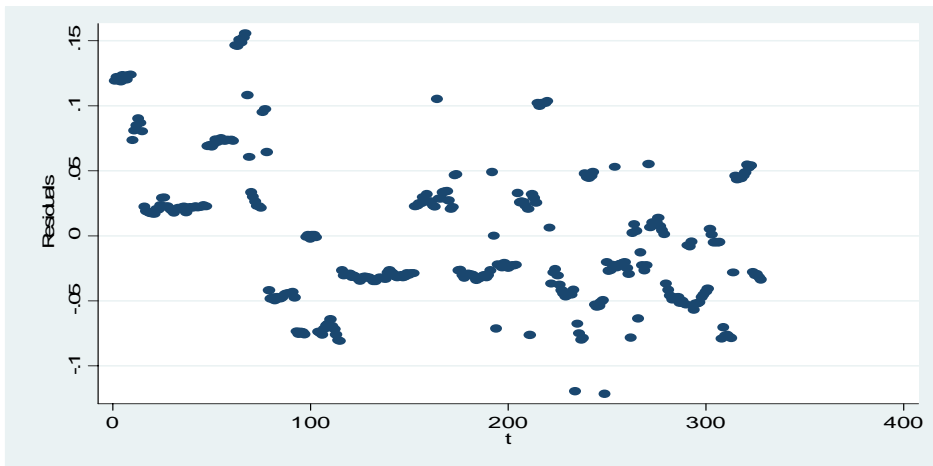
. reg price cartel quantity              \* OLS regression *\
```

Source	SS	df	MS	Number of obs =	328
Model	.491785936	2	.245892968	F(2, 325) =	83.64
Residual	.95543206	325	.002939791	Prob > F =	0.0000
-----				R-squared =	0.3398
Total	1.447218	327	.004425743	Adj R-squared =	0.3358
-----				Root MSE =	.05422

price	Coef.	Std. Err.	t	P> t	[95% Conf. Interval]	
cartel	.075482	.0064859	11.64	0.000	.0627224	.0882416
quantity	-4.73e-07	2.71e-07	-1.75	0.082	-1.01e-06	6.03e-08
_cons	.2117912	.0094683	22.37	0.000	.1931643	.2304181

```
. predict res,r                          \* generate the residuals and put them in the vector res *\

. twoway scatter res t                    \* residual plot against the index *\
```



We see a downward sloping trend in the residual terms and should suspect the existence of autocorrelation. To be sure, let's do a Durbin-Watson test.

```
. dwstat                                     \* do a Durbin-Watson test *\
```

```
Durbin-Watson d-statistic( 3, 328) = .2588336
```

This d statistic is less than the lower bound critical value in the Durbin-Watson table, so we reject the null hypothesis of no correlations. Let's do the following to correct for the autocorrelation and implement an AR(1) regression.

```
. prais price cartel quantity, corc
```

```
Iteration 0: rho = 0.0000
Iteration 1: rho = 0.8636
Iteration 2: rho = 0.9393
Iteration 3: rho = 0.9415
Iteration 4: rho = 0.9416
Iteration 5: rho = 0.9416
Iteration 6: rho = 0.9416
```

Cochrane-Orcutt AR(1) regression -- iterated estimates

Source	SS	df	MS	Number of obs = 327		
Model	.000931499	2	.00046575	F(2, 324) =	1.14	
Residual	.132029788	324	.000407499	Prob > F	= 0.3202	
Total	.132961288	326	.000407857	R-squared	= 0.0070	
				Adj R-squared	= 0.0009	
				Root MSE	= .02019	

price	Coef.	Std. Err.	t	P> t	[95% Conf. Interval]	
cartel	.0064046	.0044085	1.45	0.147	-.0022684	.0150775
quantity	-6.45e-08	1.68e-07	-0.38	0.701	-3.94e-07	2.65e-07
_cons	.2362932	.0197794	11.95	0.000	.197381	.2752054

rho	.9416093
-----	----------

```
Durbin-Watson statistic (original)    0.258834
Durbin-Watson statistic (transformed) 1.987670
```

Now, the Durbin-Watson statistic is 1.987670 which falls in the inconclusive range and the correlation coefficient is given by 0.9416093.

Next, we touch a little bit about model selection by obtaining the AIC and BIC values.

```
. reg price cartel quantity                                \* OLS regression *\
```

Source	SS	df	MS	Number of obs =	328
Model	.491785936	2	.245892968	F(2, 325) =	83.64
Residual	.95543206	325	.002939791	Prob > F =	0.0000
				R-squared =	0.3398
				Adj R-squared =	0.3358
Total	1.447218	327	.004425743	Root MSE =	.05422

price	Coef.	Std. Err.	t	P> t	[95% Conf. Interval]
cartel	.075482	.0064859	11.64	0.000	.0627224 .0882416
quantity	-4.73e-07	2.71e-07	-1.75	0.082	-1.01e-06 6.03e-08
_cons	.2117912	.0094683	22.37	0.000	.1931643 .2304181

```
. estat ic                                                \* give the AIC and BIC statistics *\
```

Model	Obs	ll(null)	ll(model)	df	AIC	BIC
.	328	424.0209	492.1194	3	-978.2388	-966.8598

Usually, we choose models with small AIC.

One-way and Two-way Anova

```
. cd z:\                                                  \* enter the working directory *\
z:\
```

```
. insheet using anova.csv                                \* open the anova data set *\
(4 vars, 32 obs)
```

```
. table a, contents(mean y)                             \* display the averages of y for each group in a *\
```

A	mean(y)
1	5.6875
2	5.0625

```
. table b, contents(mean y)                             \* display the averages of y for each group in b *\
```

B	mean(y)
1	2.75
2	3.5
3	6.25
4	9

. oneway y a, tabulate

* one-way anova of y and a *\

A	Summary of Y		Freq.
	Mean	Std. Dev.	
1	5.6875	2.120338	16
2	5.0625	3.3159966	16
Total	5.375	2.7562246	32

Source	Analysis of Variance			F	Prob > F
	SS	df	MS		
Between groups	3.125	1	3.125	0.40	0.5301
Within groups	232.375	30	7.74583333		
Total	235.5	31	7.59677419		

Bartlett's test for equal variances: $\chi^2(1) = 2.8109$ Prob> $\chi^2 = 0.094$

. oneway y b, tabulate

* one-way anova of y and b *\

B	Summary of Y		Freq.
	Mean	Std. Dev.	
1	2.75	1.4880476	8
2	3.5	.9258201	8
3	6.25	1.0350983	8
4	9	1.3093073	8
Total	5.375	2.7562246	32

Source	Analysis of Variance			F	Prob > F
	SS	df	MS		
Between groups	194.5	3	64.83333333	44.28	0.0000
Within groups	41	28	1.46428571		
Total	235.5	31	7.59677419		

Bartlett's test for equal variances: $\chi^2(3) = 1.8281$ Prob> $\chi^2 = 0.609$

. anova y b

* anova analysis of y and b *\

Number of obs = 32 R-squared = 0.8259
 Root MSE = 1.21008 Adj R-squared = 0.8072

Source	Partial SS	df	MS	F	Prob > F
Model	194.5	3	64.83333333	44.28	0.0000
b	194.5	3	64.83333333	44.28	0.0000
Residual	41	28	1.46428571		

```

-----+-----
Total |      235.5   31  7.59677419

. anovacontrast b, values(1 1 -2 0) \* test the mean of y in both group 1 and 2 is
equal to the mean of y in group 3 *\

```

```

Contrast variable: b (1 1 -2 0) Dep Var: y
source      SS      df      MS      Contrast =      -6.25
-----+-----
contrast | 52.0833333      1      52.0833      N of obs =      32
error    |      41      28      1.4643      F =      35.57
-----+-----
Prob > F =      0.0000
t =      5.96

```

```

. anovacontrast b, values(0 0 1 -1) \* test the mean of y in group 3 is equal to that in
group 4 *\

```

```

Contrast variable: b (0 0 1 -1) Dep Var: y
source      SS      df      MS      Contrast =      -2.75
-----+-----
contrast |      30.25      1      30.2500      N of obs =      32
error    |      41      28      1.4643      F =      20.66
-----+-----
Prob > F =      0.0001
t =      4.55

```

```

. reg y a \* OLS regression of y on the binary variable a *\

```

```

Source |      SS      df      MS      Number of obs =      32
-----+-----
Model  |      3.125      1      3.125      F( 1, 30) =      0.40
Residual | 232.375      30  7.74583333      Prob > F =      0.5301
-----+-----
Total  |      235.5      31  7.59677419      R-squared =      0.0133
Adj R-squared = -0.0196
Root MSE =      2.7831

```

```

-----+-----
y |      Coef.      Std. Err.      t      P>|t|      [95% Conf. Interval]
-----+-----
a |      -.625      .9839864      -0.64      0.530      -2.634568      1.384568
_cons |      6.3125      1.555819      4.06      0.000      3.135094      9.489906
-----+-----

```

In this regression, the mean of y in group 1 is given by $6.3125 - 0.625 = 5.6875$. And the mean of y in group 2 can't be inferred from the coefficients.

```

. reg y a1 \* OLS regression of y on the binary variable a1 *\

```

```

Source |      SS      df      MS      Number of obs =      32
-----+-----
Model  |      3.125      1      3.125      F( 1, 30) =      0.40
Residual | 232.375      30  7.74583333      Prob > F =      0.5301
-----+-----
Total  |      235.5      31  7.59677419      R-squared =      0.0133
Adj R-squared = -0.0196
Root MSE =      2.7831

```

```

-----+-----
y |      Coef.      Std. Err.      t      P>|t|      [95% Conf. Interval]
-----+-----

```

a1	-.625	.9839864	-0.64	0.530	-2.634568	1.384568
_cons	5.6875	.6957834	8.17	0.000	4.266521	7.108479

Different from the previous model, a1 is now a (0,1) binary variable. So from the coefficient estimates in the table, we can get the mean of y in both groups. The mean of y in group 0 is 5.6875 and the mean of y in group 1 is 0.625 less, which is 5.6875-0.625 = 5.0625.

```
. anova y a b \* anova analysis of only the main effects *\
```

```
Number of obs = 32 R-squared = 0.8392
Root MSE = 1.18439 Adj R-squared = 0.8153
```

Source	Partial SS	df	MS	F	Prob > F
Model	197.625	4	49.40625	35.22	0.0000
a	3.125	1	3.125	2.23	0.1471
b	194.5	3	64.8333333	46.22	0.0000
Residual	37.875	27	1.40277778		
Total	235.5	31	7.59677419		

```
. anova y a b a*b \* anova analysis of both the main effects and the interaction effect *\
```

```
Number of obs = 32 R-squared = 0.9214
Root MSE = .877971 Adj R-squared = 0.8985
```

Source	Partial SS	df	MS	F	Prob > F
Model	217	7	31	40.22	0.0000
a	3.125	1	3.125	4.05	0.0554
b	194.5	3	64.8333333	84.11	0.0000
a*b	19.375	3	6.45833333	8.38	0.0006
Residual	18.5	24	.770833333		
Total	235.5	31	7.59677419		

And we can tell from the table that the interaction effect is statistically significant.

Multicollinearity

Perfect multicollinearity (design matrix X is singular) affects interpretability, causes unreliable predictions for future observations.

In case of high multicollinearity (not perfect but in the range of around 0.9), we are not going to be able to understand how various predictors impact the dependent variable, such as individual P-values might be misleading, confidence intervals will be quite wide and adding or deleting a single data point might change the coefficients dramatically, or in other words, we can have very unstable coefficient estimates.

Let's start with a constructed highly correlated data set.

```
. matrix a=(1,0.98\0.98,1)          \* construct a 2X2 matrix a *\
. corr2data x1 x2, n(100) corr(a)    \* generate variables x1 and x2 with 100
observations and correlation coefficient defined in matrix a which is 0.98 *\
. gen y =1.3+0.5*x1+0.05*x2+invnorm(uniform()) \* generate y = 1.3+0.5x1+0.05x2+e
where e~normal(0,1) *\
. reg y x1 x2                        \* OLS regression of y on x1 and x2 *\
```

Source	SS	df	MS			
Model	25.6160633	2	12.8080316	Number of obs =	100	
Residual	82.240528	97	.847840495	F(2, 97) =	15.11	
Total	107.856591	99	1.08946052	Prob > F =	0.0000	
				R-squared =	0.2375	
				Adj R-squared =	0.2218	
				Root MSE =	.92078	

y	Coef.	Std. Err.	t	P> t	[95% Conf. Interval]	
x1	.1589979	.4650417	0.34	0.733	-.7639811	1.081977
x2	.3518699	.4650417	0.76	0.451	-.5711091	1.274849
_cons	1.231142	.0920783	13.37	0.000	1.048392	1.413892

```
. cor x1 x2                          \* display the correlation of x1 and x2 *\
(obs=100)
```

	x1	x2
x1	1.0000	
x2	0.9800	1.0000

```
. vif                                  \* calculate the variance inflation factors of x1 and x2 *\
```

Variable	VIF	1/VIF
x1	25.25	0.039600
x2	25.25	0.039600
Mean VIF	25.25	

If vif > 10, we know there exists multicollinearity.

```
. reg y x1                            \* OLS regression of y only on x1 *\
```

Source	SS	df	MS			
Model	25.130669	1	25.130669	Number of obs =	100	
Residual	82.7259223	98	.844142065	F(1, 98) =	29.77	
Total	107.856591	99	1.08946052	Prob > F =	0.0000	
				R-squared =	0.2330	
				Adj R-squared =	0.2252	
				Root MSE =	.91877	

y	Coef.	Std. Err.	t	P> t	[95% Conf. Interval]	

	x1					
	.5038305	.0923401	5.46	0.000	.3205846	.6870763
_cons	1.231142	.0918772	13.40	0.000	1.048814	1.413469

Let's look at a real data set next.

```
. clear
. insheet using bodyfat.csv          \* read the bodyfat dataset *\
(5 vars, 20 obs)

. reg bodyfat triceps thigh midarm \* OLS regression of bodyfat on the size of
triceps, the size of thigh and the size of midarm *\
```

Source	SS	df	MS	Number of obs =	20
Model	396.984607	3	132.328202	F(3, 16) =	21.52
Residual	98.4049068	16	6.15030667	Prob > F =	0.0000
				R-squared =	0.8014
				Adj R-squared =	0.7641
Total	495.389513	19	26.0731323	Root MSE =	2.48

bodyfat	Coef.	Std. Err.	t	P> t	[95% Conf. Interval]
triceps	4.334085	3.015511	1.44	0.170	-2.058512 10.72668
thigh	-2.856842	2.582015	-1.11	0.285	-8.330468 2.616785
midarm	-2.186056	1.595499	-1.37	0.190	-5.568362 1.19625
_cons	117.0844	99.78238	1.17	0.258	-94.44474 328.6136

```
. cor triceps thigh midarm          \* calculate the correlations *\
(obs=20)
```

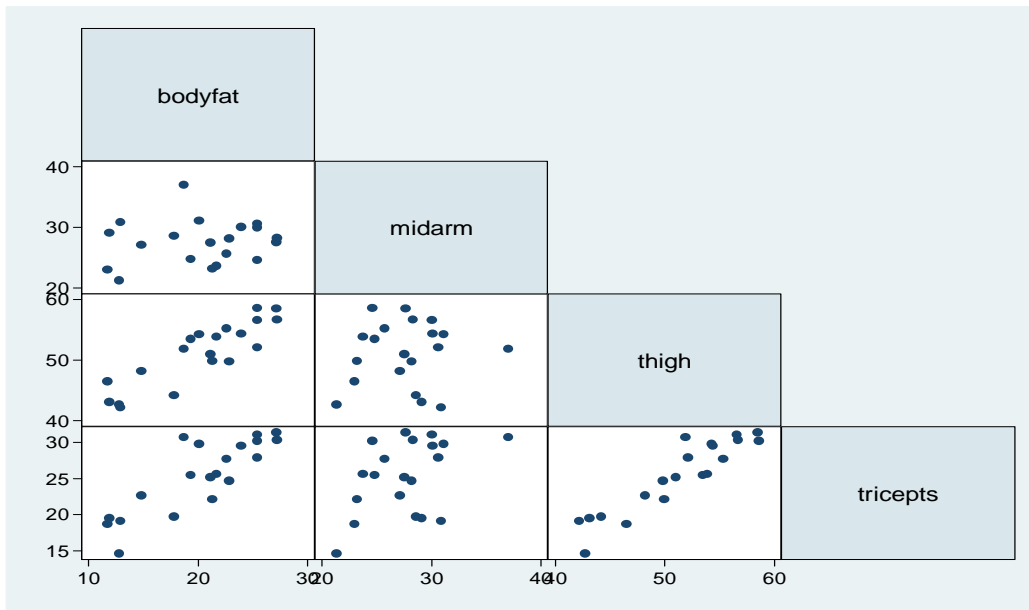
	triceps	thigh	midarm
triceps	1.0000		
thigh	0.9238	1.0000	
midarm	0.4578	0.0847	1.0000

It's very obvious that there is high correlation (0.9238) between the size of triceps and that of the thigh.

```
. vif          \* calculate the vif of these three variables *\
```

Variable	VIF	1/VIF
triceps	708.84	0.001411
thigh	564.34	0.001772
midarm	104.61	0.009560
Mean VIF	459.26	

```
. graph matrix bodyfat midarm thigh triceps, half \* scatter plots among any two of
the variables *\
```



From the scatter plot, we can also see a clear linear correlation between triceps and thigh.

```
. reg bodyfat thigh midarm \* OLS regression after dropping triceps *\
```

Source	SS	df	MS	Number of obs =	20
Model	384.279748	2	192.139874	F(2, 17) =	29.40
Residual	111.109765	17	6.53586854	Prob > F =	0.0000
Total	495.389513	19	26.0731323	R-squared =	0.7757
				Adj R-squared =	0.7493
				Root MSE =	2.5565

bodyfat	Coef.	Std. Err.	t	P> t	[95% Conf. Interval]
thigh	.8508818	.1124482	7.57	0.000	.6136367 1.088127
midarm	.0960295	.1613927	0.60	0.560	-.2444792 .4365383
_cons	-25.99696	6.99732	-3.72	0.002	-40.76001 -11.2339

```
. vif \* calculate vif of midarm and thigh *\
```

Variable	VIF	1/VIF
midarm	1.01	0.992831
thigh	1.01	0.992831
Mean VIF	1.01	

Now $vif < 10$ and since $1/vif$ is very close to 1, there is no multicollinearity problem in the model with triceps dropped.

Although dropping one or some of the highly correlated variables can remove multicollinearity, the biasness problem might arise. In practice, people still drop variables in spite of the biasness problem because prediction accuracy could be improved by sacrificing a bit of bias in exchange for smaller variance.