STATA CRASH COURSE

BASIC SYNTAX (基本語法)

Christy Pu

健康政策計量分析

2023

Today's objective

- 熟悉STATA介面，了解基本語法 (再來就可獨立學習，Google是你的好朋友)。
- 實際操作簡單的健保資料。

- I will assume:
- You already have SAS/Excel/SPSS-readable files.
- You have basic knowledge on statistics (what is a t-test, chi-2, logistic regression…etc)

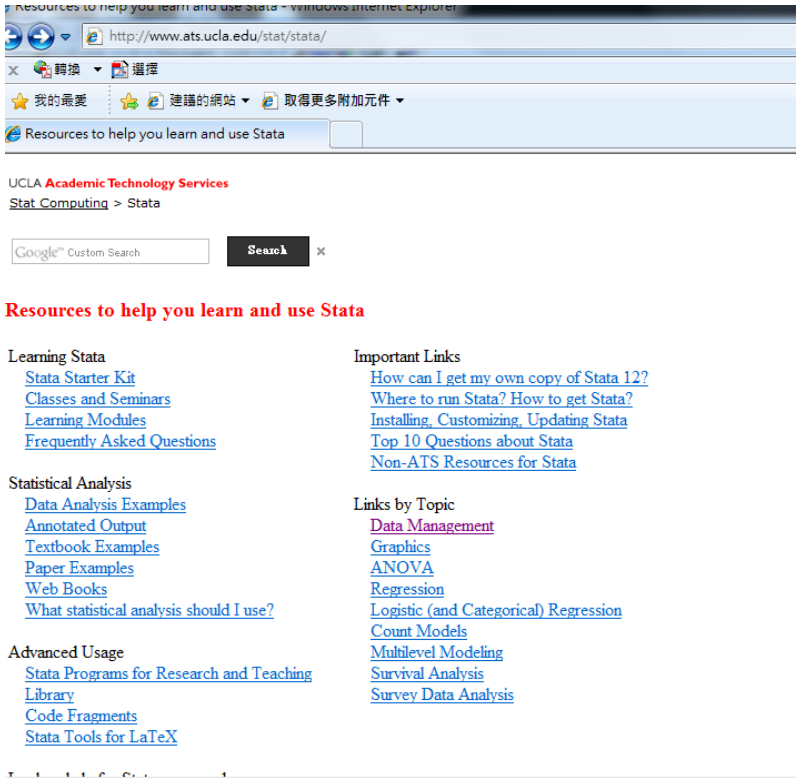
Why STATA?

- Why not? There is nothing other programs can do while STATA can't.
- 分析比SAS快很多、程式好寫很多
- Misconception: STATA can't handle large datasets → This depends on your hardware, not on STATA.
- Can STATA handle NHI data with longitudinal calculation? Definitely!

- However: There may be certain programs that do run slower in STATA compared with other statistical software's (e.g. multilevel is kind of slow; machine learning is better with R)
- Personal experience: Nothing serious (thesis, dissertation, publications with complicated models)

Resources

- Make friends with this website: http://www.ats.ucla.edu/stat/stata/



Resources

- If you are unsure of a command, always Google it. Most of the times it leads you directly you to the UCLA websites, or STATA's own FAQ.

e.g.: just Google "STATA merge"

- STATA interface
- “findit” → e.g findit merge
- The best way to learn STATA (and any software) is by typing in the commands.
- Definition of how good you can handle a statistical program: how many commands you know and remember.
- No single course can teach you all.
- **Just play with it!**

- compress →Type “compress” and STATA will store the variables in the most efficient way for you.

- clear → whenever you type “clear”, STATA will empty the memory. Note if you have unsaved work you will lose it too.

Getting started

- Do-file (always keep a do-file → record every single command you use)
- 請保持寫程式的良好習慣: the first command in every do-file should be defining the dataset you use.
- use "C:\Users\Christy\Desktop\STATA Summer Course\Car.dta", clear
- Note there is maximum limit of 131071 bytes for a do-file.
- (not a problem unless you have bad programming habits)
- Log file

- log using "C:\Users\Christy\Desktop\STATA Summer Course\Car" , replace
- xxxxxx
- yyyy
- log close

Inputting data

- Copy and paste.
  - Excel: The first line in the spreadsheet should have the variable names
- StatTansfer → open directly
- Transfer it using SAS

Inputting data

- use "C:\Users\Christy\Desktop\STATA Summer Course\Car.dta" , clear

- The clear option will clear the revised dataset currently in memory before opening the other one.
- Basic notation and rules:
  - A variable can not start with number, e.g. 2005age (use age2005, _2005age instead)
  - Underscore(底線) → hyphen will be read as “minus” or ”to”
  - Always put “ “ (引號) for string variables. E.g count if name==“Mark”
  - Note: font matter when you copy and paste from, say, PPT.
  - gen aa=“cccc”

- Missing data in Stata is “.” (for numeric) and “ ” (for string) Thus “…” “-” “999” etc. will be read as they are. You can always replace them later using commands.
- E.g. replace sex=. If sex==999

- Note different versions handle capital/common letters differently, but they matters (弄錯會出現紅字喔!)
- However, capital letters is allowed for variables.
- All commands are in common letters (弄錯會出現紅字喔!)
- .

Useful notations

- Equal (make changes): = → e.g generate year=2005
- Equal (identity) == → e.g. count if year==2005

```
. count if price=5000
=exp not allowed
r(101);

. count if make="Ford"
type mismatch
r(109);
```

- Not equal to: ~=
- > , <, >=, <= +(加), - (減), *(乘) /(除)
- and: & → count if name==“Mark” & age==20
- or: | → count if name==“Mark”|name==“John”
- count if (name==“Mark”|name==“John”) & age==20

In-class exercise

- Open the course file car.dta.
- Open a new do-file, save it
- 總共有幾筆資料? (答案:74)
- 計算: 平均價格是多少? (答案: 6165.3)
- 把variable “make” 改名成 “manufacturer”
- 有多少車是foreign-made? (variable name: foreign) (答案:22)
- 有多少車價格>=5000? (答案:37)

Useful commands

- Attaches a label (up to 80 characters) to the dataset →
  - label data “1978 Automobile Data, In-class exercise”
- Label variable (or label var) →
  - label var price “price of car”
    - label variable price “” (to clear)
- Label define →
  - label define stock 0 no 1 yes 2 maybe
  - label list stock
- label drop → label drop stock
- Rename → rename stock instock

Examining data

```
. list price length in 45/49

     |  price   length |
     |-----------------|
45.  | 6,486      182  |
46.  | 4,060      201  |
47.  | 5,798      214  |
48.  | 4,934      198  |
49.  | 5,222      201  |
     +-----------------+
```

- *list price*
- *list price if price>6000*
- *list in 45/49 (list only the 45~49 observations)*

- *sum (short for summarize, only for numeric values)*
- *sum price*
- *sum price, detail*
- *sum price if length<200*
- *sum price if weight>3000*
- *sum price if weight>3000, detail*

```
. sum price weight headroom if length<200
```

Variable	Obs	Mean	Std. Dev.	Min	Max
price	47	5318.489	2092.824	3291	12990
weight	47	2567.021	533.2261	1760	3470
headroom	47	2.670213	.7243933	1.5	4.5

```
. sum price if length<200, detail
```

price of the car

Percentiles		Smallest		
1%	3291	3291		
5%	3667	3299		
10%	3798	3667	Obs	47
25%	4099	3748	Sum of Wgt.	47
50%	4589		Mean	5318.489
		Largest	Std. Dev.	2092.824
75%	5799	9690		
90%	8129	9735	Variance	4379912
95%	9735	11995	Skewness	2.146823
99%	12990	12990	Kurtosis	7.373537

tabstat

- tabstat price weight mpg
- tabstat price weight mpg, by(foreign) stat(mean sd min max) long

-

mean	mean
count	count of nonmissing
n	same as count
sum	sum
max	maximum
min	minimum
range	range = max - min
sd	standard deviation
variance	variance
cv	coefficient of variation (sd/mean)
semean	standard error of mean
skewness	skewness
kurtosis	kurtosis
p1	1st percentile
p5	5th percentile
p10	10th percentile
p25	25th percentile
median	median (same as p50)
p50	50th percentile (same as median)
p75	75th percentile
p90	90th percentile
p95	95th percentile
p99	99th percentile
iqr	interquartile range = p75 - p25
q	equivalent to specifying p25 p50 p75

```
. tabstat price weight mpg, by(foreign) stat(mean sd min max)  long
```

foreign	stats	price	weight	mpg
Domestic	mean	6072.423	3317.115	19.82692
	sd	3097.104	695.3637	4.743297
	min	3291	1800	12
	max	15906	4840	34
Foreign	mean	6384.682	2315.909	24.77273
	sd	2621.915	433.0035	6.611187
	min	3748	1760	14
	max	12990	3420	41
Total	mean	6165.257	3019.459	21.2973
	sd	2949.496	777.1936	5.785503
	min	3291	1760	12
	max	15906	4840	41

Examining data

- tab length
- tab length if length<170
- tab length if length<170, sum(weight)

- 保持良好習慣: 跑迴歸前先sum或tab所有變項，並作完整描述性統計。

- **Saving the dataset**
- save "C:\Users\Christy NB\Desktop\STATA Summer Course\Car.dta", replace
- (即使是全新檔名，也可以打replace)

```
. tab length if length<170
```

Length (in.)	Freq.	Percent	Cum.
142	1	5.88	5.88
147	1	5.88	11.76
149	1	5.88	17.65
154	1	5.88	23.53
155	2	11.76	35.29
156	1	5.88	41.18
157	1	5.88	47.06
161	1	5.88	52.94
163	2	11.76	64.71
164	1	5.88	70.59
165	3	17.65	88.24
168	1	5.88	94.12
169	1	5.88	100.00
Total	17	100.00	

```
. tab length if length<170, sum(weight)
```

Length (in.)	Summary of Weight (lbs.) Mean	Std. Dev.	Freq.
142	1,830	0	1
147	1,800	0	1
149	1,760	0	1
154	1,980	0	1
155	1,985	77.781746	2
156	1,990	0	1
157	1,800	0	1
161	2,130	0	1
163	2,115	7.0710678	2
164	2,050	0	1
165	2,140	103.92305	3
168	2,640	0	1
169	2,580	0	1
Total	2,069.412	245.44527	17

Basic operations

- Keep variable(s) →
  - keep price weight length
  - Keep christy* (keep all variables beginning with christy)
- Drop variable(s) → drop price weight
- Generate a variable →
  - gen want=1 if price<=4500 & stock==1
  - gen eco = length + weight

```
. gen eco = length + weight if price<6000
(23 missing values generated)

. gen eco=length/weight if price<6000
eco already defined
r(110);
```

- 保持良好習慣: always drop those variables you mistakenly generated.

- For string variables →
- gen car_id=[own] + [id]
- gen car_id2 = [own] + " loves STATA" if own=="Christy"
- replace car_id2 = [own] + " loves SAS" if own=="Hu"

- Replace →
- replace stock=. if price>7000
- replace eco= height/weight

“gen” vs. “egen”

- “gen” is to generate a variable.
- “egen” generates a variable containing a function.
- For example, I want to generate a variable containing the mean of *price*:

- egen mean_price = mean(price)
- You can then do whatever calculation you want, e.g:

count if price > mean_price

keep if price > mean_price

	price	mean_price
1	4,099	6165.257
2	4,749	6165.257
3	3,799	6165.257
4	4,816	6165.257
5	7,827	6165.257
6	5,788	6165.257
7	4,453	6165.257
8	5,189	6165.257
9	10,372	6165.257
10	4,082	6165.257
11	11,385	6165.257
12	14,500	6165.257
13	15,906	6165.257
14	3,299	6165.257
15	5,705	6165.257
16	4,504	6165.257
17	5,104	6165.257
18	3,667	6165.257
19	3,955	6165.257
20	3,984	6165.257

egen

- Functions in egen include:
- mean, sd, max, mean, median, mode, rank, iqr, kurt, skew, rowmean, rowmax…etc

- For more functions: findit egen

- For percentiles and quantiles:
- pctile p_price = price, nq(5)
- and
- xtile x_price = price, nq(5)

	price	mean_price	p_price	x_price
1	4,099	6165.257	4099	1
2	4,749	6165.257	4647	3
3	3,799	6165.257	5705	1
4	4,816	6165.257	7827	3
5	7,827	6165.257	.	4
6	5,788	6165.257	.	4
7	4,453	6165.257	.	2
8	5,189	6165.257	.	3
9	10,372	6165.257	.	5
10	4,082	6165.257	.	1
11	11,385	6165.257	.	5
12	14,500	6165.257	.	5
13	15,906	6165.257	.	5
14	3,299	6165.257	.	1
15	5,705	6165.257	.	3
16	4,504	6165.257	.	2
17	5,104	6165.257	.	3
18	3,667	6165.257	.	1
19	3,955	6165.257	.	1
20	3,984	6165.257	.	1
21	4,010	6165.257		1

“sort” and “by”

- Sort in ascending order → sort
- E.g. →
- sort id
- sort id sex

- Sort in descending order:
- gsort id -sex (the minus sign means this variable is to be sorted in descending order)
- Sort must be performed before “by” →
- sort own
- by own: sum price
- by own: logistic Y X_1 X_2 X_3

```
sort own

by own: sum price
```

-> own = Christy

Variable	Obs	Mean	Std. Dev.	Min	Max
price	16	5317.375	2225.208	3798	12990

-> own = Hu

Variable	Obs	Mean	Std. Dev.	Min	Max
price	21	6409.762	2998.587	3291	14500

-> own = Jack

Variable	Obs	Mean	Std. Dev.	Min	Max
price	22	6626.773	3229.574	3799	15906

Merge files

Data Editor (Edit) - [master]

File Edit View Data Tools

var7[14]

	id	income
1	1	10
2	2	20
3	3	30
4	4	.
5	5	50
6	6	60
7	7	.
8	8	80
9	9	90
10	10	100

Master file

+

Data Editor (Edit) - [using]

File Edit View Data Tools

id[1] 1

	id	income	sex
1	1	10	1
2	2	100	2
3	3	20	2
4	4	200	1
5	5	300	2
6	6	30	1
7	7	40	2
8	8	400	1
9	9	500	2
10	10	50	1
11	11	70	1
12	12	80	2
13	13	90	1
14	14	90	2
15	15	80	1

Using file

Merge (con't)

- merge 1:1 id using "C:\Users\Christy\Desktop\using.dta"

- _merge

1 → observation only appears in the Master data
2 → observation only appears in the Using data
3 → Observations in both Master and Using data

merge 1:1 id using "C:\Users\Christy\Desktop\using.dta", update

STATA will use the Using values only when Master has missing values.

merge 1:1 id using "C:\Users\Christy\Desktop\using.dta", replace update

- STATA will use the value in Using to replace Master.

merge 1:1 id using "C:\Users\Christy\Desktop\using.dta", update

	id	income	sex	_merge
1	1	10	1	matched (3)
2	2	20	2	nonmissing conflict (5)
3	3	30	2	nonmissing conflict (5)
4	4	200	1	missing updated (4)
5	5	50	2	nonmissing conflict (5)
6	6	60	1	nonmissing conflict (5)
7	7	40	2	missing updated (4)
8	8	80	1	nonmissing conflict (5)
9	9	90	2	nonmissing conflict (5)
10	10	100	1	nonmissing conflict (5)
11	11	70	1	using only (2)
12	12	80	2	using only (2)
13	13	90	1	using only (2)
14	14	90	2	using only (2)
15	15	80	1	using only (2)

```
. merge 1:1 id using  "C:\Users\Christy\Desktop\using.dta", update

    Result                           # of obs.
    -----------------------------------------
    not matched                             5
        from master                         0  (_merge==1)
        from using                          5  (_merge==2)

    matched                                10
        not updated                         1  (_merge==3)
        missing updated                     2  (_merge==4)
        nonmissing conflict                 7  (_merge==5)
    -----------------------------------------

. tab _merge

                _merge |      Freq.     Percent        Cum.
-----------------------+-----------------------------------
        using only (2) |          5       33.33       33.33
            matched (3) |          1        6.67       40.00
    missing updated (4) |          2       13.33       53.33
nonmissing conflict (5) |          7       46.67      100.00
-----------------------+-----------------------------------
                 Total |         15      100.00
```

Merge files (con't)

保持良好習慣: merge前先預測應出現什麼數值，然後 tab _merge 檢察

Q: 健保資料庫ID檔=master,
CD檔=using,
merge後應該會出現哪些 _merge值?

Always drop _merge before performing next merge. drop _merge

```
. merge id using  "C:\Users\Christy NB\Downloads\bb.dta"
_merge already defined
r(110);
```

aa

	id	marital	sex	Ces_D	year
1	a	S	M	3	1995
2	b	M	M	2	1995
3	c	M	F	4	1995
4	d	S	M	10	1995
5	e	S	M	2	1995

+

bb

	id	marital	sex	Ces_D	year
1	a	S	M	0	1996
2	b	M	M	5	1996
3	c	D	F	9	1996
4	d	S	M	25	1996
5	e	D	M	18	1996

=

	id	marital	sex	Ces_D	year
1	a	S	M	3	1995
2	b	M	M	2	1995
3	c	M	F	4	1995
4	d	S	M	10	1995
5	e	S	M	2	1995

This is definitely not what we wanted.

Merge is used to add variables, what if we want to add observations?

Answer: use *append*

Append

Option1 : rename your variables
Option 2: use “append”

append using "C:\Users\Christy NB\Desktop\bb.dta"

	id	marital	sex	Ces_D	year
1	a	S	M	3	1995
2	b	M	M	2	1995
3	c	M	F	4	1995
4	d	S	M	10	1995
5	e	S	M	2	1995
6	a	S	M	0	1996
7	b	M	M	5	1996
8	c	D	F	9	1996
9	d	S	M	25	1996
10	e	D	M	18	1996

Working with dates in STATA

We will learn 3 things about date in STATA:

- Creating new (clean) date variables from your old (dirty) date variable using the “date” function.
- Formatting the new date variable so it displays in a way that you can read it as a date.
- Ways to use STATA’s special date functions to analyze date variables.

- STATA calculates days elapsed from Jan 1, 1960.
- So Jan 2, 1960 is represented by the number “1”, and Jan 1, 1961 by the number “365”.

- generate date1=date(car_date, “YMD")
- format date1 %d

	car_date
1	20010101
2	19950301
3	20050617
4	20051007
5	20040503
6	19990701
7	20030721
8	20051115
9	19950301
10	20020905
11	19950301
12	19950301
13	20010726
14	20020610
15	20041130

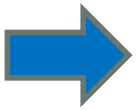
	car_date	date1
1	20010101	14976
2	19950301	12843
3	20050617	16604
4	20051007	16716
5	20040503	16194
6	19990701	14426
7	20030721	15907
8	20051115	16755
9	19950301	12843
10	20020905	15588
11	19950301	12843
12	19950301	12843
13	20010726	15182
14	20020610	15501
15	20041130	16405

	id	car_date	date1
1	var14	20010101	01jan2001
2	ddd	19950301	01mar1995
3	bbb	20050617	17jun2005
4	bbb	20051007	07oct2005
5	bbb	20040503	03may2004
6	ppp	19990701	01jul1999
7	ddd	20030721	21jul2003
8	aaa	20051115	15nov2005
9	var14	19950301	01mar1995
10	zzz	20020905	05sep2002
11	aaa	19950301	01mar1995
12	bbb	19950301	01mar1995
13	bbb	20010726	26jul2001
14	zzz	20020610	10jun2002
15	xyz	20041130	30nov2004

- (Note YMD can be changed to DMY YDM etc...any order you need)

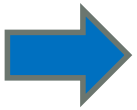
- The “date” function in STATA is very powerful, it recognize almost any date format:

name	bday
John	Jan 1 1960
Mary	07/11/1955
Kate	11.12.1962
Mark	Jun/8 1959



name	bday	birthday
John	Jan 1 1960	01jan1960
Mary	07/11/1955	11jul1955
Kate	11.12.1962	12nov1962
Mark	Jun/8 1959	08jun1959

bday b
4-12-1990
4.12.1990
Apr 12, 1990
Apr12,1990
April 12, 1990
4/12.1990
Apr121990

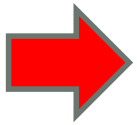


bday	birthday
4-12-1990	12apr1990
4.12.1990	12apr1990
Apr 12, 1990	12apr1990
Apr12,1990	12apr1990
April 12, 1990	12apr1990
4/12.1990	12apr1990
Apr121990 .	

Date - variations

- generate birthday=MDY (month,day,year)
- format birthday %d

month	day	year
7	11	1948
1	1	1960
10	15	1970
12	10	1971



Month	day	year	birthday
7	11	1948	11jul1948
1	1	1960	01jan1960
10	15	1970	15oct1970
12	10	1971	10dec1971

- Note that if your year is in two digits, use:
- replace birthday=MDY(month,day,year+1900)

Date - calculation

- You can do any sort of mathematical manipulation with this integer variable, like this:
- gen bday10 = birthday + 10
- 03jan1960 → 13jan1960
- gen difference = date1 – date2
- Suppose date 1 = 01jan2000, and date2 = 05jan2000,
- difference → 4

Sample birthday program in NHI

- generate birthday=date(id_birthday, "YMD")
- format birthday %d
- gen date1="20050101"
- generate date2=date(date1, "YMD")
- format date2 %d
- gen age2005=(date2-birthday)/365.25

Date -calculation

- To compare date →
- generate after1999 = 0
- replace after1999 = 1 if date1 > d(1jan1999)

- To extract dates →
- generate m=month(date1)
- generate d=day(date1)
- generate y=year(date1)
- generate w=week(date1)
- generate dw=dow(date1)

	date1	m	d	y	w	dw
1	01mar1995	3	1	1995	9	3
2	07oct2005	10	7	2005	40	5
3	03may2004	5	3	2004	18	1
4	01jul1999	7	1	1999	26	4
5	21jul2003	7	21	2003	29	1
6	15nov2005	11	15	2005	46	2
7	01mar1995	3	1	1995	9	3
8	05sep2002	9	5	2002	36	4
9	01mar1995	3	1	1995	9	3
10	01mar1995	3	1	1995	9	3
11	26jul2001	7	26	2001	30	4
12	10jun2002	6	10	2002	23	1
13	30nov2004	11	30	2004	48	2
14	10aug2001	8	10	2001	32	5
15	25jun2005	6	25	2005	26	6
16	23dec2005	12	23	2005	51	5
17	31mar2005	3	31	2005	13	4

Extract a portion of the string variable

- When will this come in handy? E.g.
- In NHI data, I want to analyze ICD-9 codes begin with 250 (this include 250XX, 250.XX, 25012345 etc)
- In NHIS data, there are two distinct cohorts stored in the same database (age 12~65 and age>65), but I just want to analyze the first cohort.
- I am analyzing open-ended questions, e.g. “Why didn’t you buy insurance?”, and this is open ended. I want to analyze those whose answers include “financial” → this include “financial reasons”, “I don’t want to buy insurance due to financial reasons” “I have financial problems!”, “I am financially limited” etc.

E.g. 1 → ICD-9 codes in NHI

- gen diabetes=1 if(regexm(icd, "^[2][5][0]"))
- or gen aa=regexs(0) if(regexm(icd, "^[2][5][0]"))
- The ^ means counting from the first letter (else 3250 will also be marked)

	icd	diabetes	aa
1	25000	1	250
2	38033	.	
3	188.00	.	
4	250.xx	1	250
5	250133	1	250
6	V3462	.	
7	250	1	250
8	250.277	1	250
9	419	.	
10	250.	1	250

- Two functions to remember: regexm and regexs
- regexm = regular expressions that matches (something)
- regexs = regular expressions specified by regexm

How regexs is represented?

- Regexm(variable, "([0-9]*)\-([0-9]*)\-([0-9]*)")
- Subexpression # String Returned
- regexs(0) 907-789-3939
- regexs(1) 907
- regexs(2) 789
- regexs(3) 3939
- E.g. gen aa= regexs(2) if regexm(variable, "([0-9]*)\-([0-9]*)\-([0-9]*)")
- Will give you aa =789.
- Tip1: Simply count from the bracket after the variable name (not including redundant brackets).
- Tip2: Try it with a few observations first, may require lots of trial and error

E.g. 1 → ICD-9 codes in NHI (con't)

- Extracts the first three codes:
- gen icd9_3=regexs(0) if(regexm(icd, "^[0-9a-zA-Z][0-9][0-9]"))

	icd	icd9_3
1	25000	250
2	38033	380
3	188.00	188
4	250.xx	250
5	250133	250
6	V3462	V34
7	250	250
8	250.277	250
9	419	419
10	250.	250

- Analyzing only those with id being with "B" in NHIS is similar.

E.g (2): get those who replied “financial”

gen financial= regexs(0) if (regexm(reasons, "[Ff][iI][nN][aA][nN][cC][iI][Aa][lL]"))

	reasons	financial
1	I am financially limited	financial
2	Financial problems	Financial
3	I hate the government	
4	Money-financiallly!!!	financial
5	Insurance is stupid	
6	None of your business	
7	FiNanCiaLXDDD :)	FiNanCiaL
8	No need	
9	I have financialissue	financial
10	I am short-sighted	

E.g. → Detect separate words

- I have first and last name in the same variable, but I want them in separate variables. → e.g., I want “Christy Pu” to be stored as “Christy” and “Pu”.
- gen Firstname = regexs(2) if regexm(name, "(([a-zA-Z]*)[ ]([a-zA-Z]*))")
- gen Lastname = regexs(3) if regexm(name, "(([a-zA-Z]*)[ ]([a-zA-Z]*))")

	name
1	Christy Pu
2	Little Bear
3	Jack Chou
4	Mark Yang
5	Pretty Hu

Firstname	Lastname
Christy	Pu
Little	Bear
Jack	Chou
Mark	Yang
Pretty	Hu

E.g → Extract the *n*th letter

- I want to get the 4th letter for each name.
- gen fourth = regexs(5) if regexm(name, "(([a-zA-Z])([a-zA-Z])([a-zA-Z])([a-zA-Z]))")

name	fourth
Christy Pu	i
Little Bear	t
Jack Chou	k
Mark Yang	k
Pretty Hu	t

String and numeric variables

- You may wish to change the variable type interchangeably.
- E.g. Sex is stored as “Male” and “Female”, you want male=1 and female = 2.
- You have a variable stored in numeric form but you want it to be in string form (and vice versa). E.g.→ date.
- (This often happens when you copy and paste variable from STATA to, say, Excel, or mistakes in entering data.)

- Two sets of commands:
- encode vs. decode
- destring vs. tostring

encode vs. decode (1)

- encode sex, gen(sex1) → change string to numeric, with label

	sex	sex1	sex2
1	Male	Male	1
2	Female	Female	0
3	Male	Male	1
4	Male	Male	1
5	Female	Female	0
6	Female	Female	0
7			

-
```
. label list sex1
sex1:
           1 Female
           2 Male
```

- You can now do any calculation with it, e.g. →
- gen sex2=sex1-1

encode vs. decode (2)

- To get it back to string:
- decode sex1, gen (sex3)

	sex	sex1	sex2	sex3
1	Male	Male	1	Male
2	Female	Female	0	Female
3	Male	Male	1	Male
4	Male	Male	1	Male
5	Female	Female	0	Female
6	Female	Female	0	Female
7		.	.	

destring vs. tostring (1)

- destring birthday, gen (birthday1)

	birthday	birthday1	birthday2
1	19800122	19800122	19800122
2	19751212	19751212	19751212
3	20050922	20050922	20050922
4	19980423	19980423	19980423
5	20080202	20080202	20080202
6	19680918	19680918	19680918

- Destring works only when the string variable is in number.
- To get it back to string
- tostring birthday1, gen(birthday2)

Collapse

- Extremely useful when analyzing NHI data.

	id_birthday	id	acode_icd9_1	acode_icd9_2	acode_icd9_3	drug_amt	t_amt	id_sex
1	19820900	012e61103b1db1b6e92ecf423de53aa5	5819	.	.	0	1202	M
2	19820921	012e61103b1db1b6e92ecf423de53aa5	250.11	.	.	0	1430	M
3	19820921	012e61103b1db1b6e92ecf423de53aa5	581	.	.	0	2290	M
4	19820921	012e61103b1db1b6e92ecf423de53aa5	250			0	2000	M
5	19820921	012e61103b1db1b6e92ecf423de53aa5	4658	250123	.	160	415	M
6	19740327	014fd9cb0edfc5b57255ee4a9e39660b	7804	7870	.	84	531	M
7	19740327	014fd9cb0edfc5b57255ee4a9e39660b	7294	.	.	425	2406	M
8	19810520	01adaa2d0a09b8e6781ebc0bcbc23d4d	7061	.	.	100	221	F
9	19810520	01adaa2d0a09b8e6781ebc0bcbc23d4d	5234	5210	.	0	1430	F
10	19810520	01adaa2d0a09b8e6781ebc0bcbc23d4d	250	.	.	100	241	F
11	19680724	025ce9868ffe69b3ea57018f64401c77	4650	4659	250.187	73	334	F

- The above is a hypothetical CD file
- I want to:
- Count how many outpatient each patient has.
- Calculate the average outpatient and drug expenditure by person.
- Find those who had any diagnose of 250 in acode_icd9(1-3)

Count how many outpatient each patient has (1)

gen count=1
collapse (sum) count, by (id id_birthday)

	id_birthday	id	acode_icd9_1	acode_icd9_2	acode_icd9_3	drug_amt	t_amt	id_sex
1	19820900	012e61103b1db1b6e92ecf423de53aa5	5819	.	.	0	1202	M
2	19820921	012e61103b1db1b6e92ecf423de53aa5	250.11	.	.	0	1430	M
3	19820921	012e61103b1db1b6e92ecf423de53aa5	581	.	.	0	2290	M
4	19820921	012e61103b1db1b6e92ecf423de53aa5	250			0	2000	M
5	19820921	012e61103b1db1b6e92ecf423de53aa5	4658	250123	.	160	415	M
6	19740327	014fd9cb0edfc5b57255ee4a9e39660b	7804	7870	.	84	531	M
7	19740327	014fd9cb0edfc5b57255ee4a9e39660b	7294	.	.	425	2406	M
8	19810520	01adaa2d0a09b8e6781ebc0bcbc23d4d	7061	.	.	100	221	F
9	19810520	01adaa2d0a09b8e6781ebc0bcbc23d4d	5234	5210	.	0	1430	F
10	19810520	01adaa2d0a09b8e6781ebc0bcbc23d4d	250	.	.	100	241	F
11	19680724	025ce9868ffe69b3ea57018f64401c77	4650	4659	250.187	73	334	F

	id_birthday	id	count
1	19820900	012e61103b1db1b6e92ecf423de53aa5	1
2	19820921	012e61103b1db1b6e92ecf423de53aa5	4
3	19740327	014fd9cb0edfc5b57255ee4a9e39660b	2
4	19810520	01adaa2d0a09b8e6781ebc0bcbc23d4d	3
5	19680724	025ce9868ffe69b3ea57018f64401c77	1

Count how many outpatient each patient has (2)

- Or alternatively:
- sort id id_birthday
- by id id_birthday: gen count= _N

	acode_icd9_1	acode_icd9_2	acode_icd9_3	drug_amt	t_amt	id_sex	newid	count
1	5819	.	.	0	1202	M	012e61103b1db1b6e92ecf423de53aa519820900	1
2	250.11	.	.	0	1430	M	012e61103b1db1b6e92ecf423de53aa519820921	4
3	581	.	.	0	2290	M	012e61103b1db1b6e92ecf423de53aa519820921	4
4	250			0	2000	M	012e61103b1db1b6e92ecf423de53aa519820921	4
5	4658	250123	.	160	415	M	012e61103b1db1b6e92ecf423de53aa519820921	4
6	7804	7870	.	84	531	M	014fd9cb0edfc5b57255ee4a9e39660b19740327	2
7	7294	.	.	425	2406	M	014fd9cb0edfc5b57255ee4a9e39660b19740327	2
8	7061	.	.	100	221	F	01adaa2d0a09b8e6781ebc0bcbc23d4d19810520	3
9	5234	5210	.	0	1430	F	01adaa2d0a09b8e6781ebc0bcbc23d4d19810520	3
10	250	.	.	100	241	F	01adaa2d0a09b8e6781ebc0bcbc23d4d19810520	3
11	4650	4659	250.187	73	334	F	025ce9868ffe69b3ea57018f64401c7719680724	1

_**N** is equal to the number of observations in the dataset except in a by command when it is equal to the total number of observations in the by-group.

Calculate the average outpatient expenditure by person.

collapse (mean) t_amt drug_amt, by (id id_birthday)

	id_birthday	id	acode_icd9_1	acode_icd9_2	acode_icd9_3	drug_amt	t_amt	id_sex
1	19820900	012e61103b1db1b6e92ecf423de53aa5	5819	.	.	0	1202	M
2	19820921	012e61103b1db1b6e92ecf423de53aa5	250.11	.	.	0	1430	M
3	19820921	012e61103b1db1b6e92ecf423de53aa5	581	.	.	0	2290	M
4	19820921	012e61103b1db1b6e92ecf423de53aa5	250			0	2000	M
5	19820921	012e61103b1db1b6e92ecf423de53aa5	4658	250123	.	160	415	M
6	19740327	014fd9cb0edfc5b57255ee4a9e39660b	7804	7870	.	84	531	M
7	19740327	014fd9cb0edfc5b57255ee4a9e39660b	7294	.	.	425	2406	M
8	19810520	01adaa2d0a09b8e6781ebc0bcbc23d4d	7061	.	.	100	221	F
9	19810520	01adaa2d0a09b8e6781ebc0bcbc23d4d	5234	5210	.	0	1430	F
10	19810520	01adaa2d0a09b8e6781ebc0bcbc23d4d	250	.	.	100	241	F
11	19680724	025ce9868ffe69b3ea57018f64401c77	4650	4659	250.187	73	334	F

	id_birthday	id	t_amt	drug_amt
1	19820900	012e61103b1db1b6e92ecf423de53aa5	1202	0
2	19820921	012e61103b1db1b6e92ecf423de53aa5	1533.75	40
3	19740327	014fd9cb0edfc5b57255ee4a9e39660b	1468.5	254.5
4	19810520	01adaa2d0a09b8e6781ebc0bcbc23d4d	630.667	66.6667
5	19680724	025ce9868ffe69b3ea57018f64401c77	334	73

Options available for collapse

- **mean** means (default)
- **median** medians
- **p1** 1st percentile
- **p2** 2nd percentile ... 3rd-49th percentiles **p50** 50th
- **sd** standard deviations
- **sum** sums
- **rawsum** sums,
- **max** maximums
- **min** minimums
- **iqr** interquartile range
- **first** first value
- **last** last value
- **firstnm** first nonmissing value
- **lastnm** last nonmissing value

etc.

Find those who had any diagnose of 250 in acode_icd9(1-3)

- gen dm=regexs(0) if (regexm(acode_icd9_1, “^[2][5][0]"))
- replace dm=regexs(0) if (regexm(acode_icd9_2, “^[2][5][0]"))
- replace dm=regexs(0) if (regexm(acode_icd9_3, “^[2][5][0]"))

	id_birthday	id	acode_icd9_1	acode_icd9_2	acode_icd9_3	drug_amt	t_amt	id_sex	dm
1	19820900	012e61103b1db1b6e92ecf423de53aa5	5819	.	.	0	1202	M	
2	19820921	012e61103b1db1b6e92ecf423de53aa5	250.11	.	.	0	1430	M	250
3	19820921	012e61103b1db1b6e92ecf423de53aa5	581	.	.	0	2290	M	
4	19820921	012e61103b1db1b6e92ecf423de53aa5	250			0	2000	M	250
5	19820921	012e61103b1db1b6e92ecf423de53aa5	4658	250123	.	160	415	M	250
6	19740327	014fd9cb0edfc5b57255ee4a9e39660b	7804	7870	.	84	531	M	
7	19740327	014fd9cb0edfc5b57255ee4a9e39660b	7294	.	.	425	2406	M	
8	19810520	01adaa2d0a09b8e6781ebc0bcbc23d4d	7061	.	.	100	221	F	
9	19810520	01adaa2d0a09b8e6781ebc0bcbc23d4d	5234	5210	.	0	1430	F	
10	19810520	01adaa2d0a09b8e6781ebc0bcbc23d4d	250	.	.	100	241	F	250
11	19680724	025ce9868ffe69b3ea57018f64401c77	4650	4659	250.187	73	334	F	250

Count how many times each person has DM outpatient

- gen dm=regexs(0) if (regexm(acode_icd9_1, "[2][5][0]"))
- replace dm=regexs(0) if (regexm(acode_icd9_2, "[2][5][0]"))
- replace dm=regexs(0) if (regexm(acode_icd9_3, "[2][5][0]"))
- gen DMcount=1 if dm~=""

	id_birthday	id	acode_icd9_1	acode_icd9_2	acode_icd9_3	drug_amt	t_amt	id_sex	dm	DMcount
1	19820900	012e61103b1db1b6e92ecf423de53aa5	5819	.	.	0	1202	M		.
2	19820921	012e61103b1db1b6e92ecf423de53aa5	250.11	.	.	0	1430	M	250	1
3	19820921	012e61103b1db1b6e92ecf423de53aa5	581	.	.	0	2290	M		.
4	19820921	012e61103b1db1b6e92ecf423de53aa5	250			0	2000	M	250	1
5	19820921	012e61103b1db1b6e92ecf423de53aa5	4658	250123	.	160	415	M	250	1
6	19740327	014fd9cb0edfc5b57255ee4a9e39660b	7804	7870	.	84	531	M		.
7	19740327	014fd9cb0edfc5b57255ee4a9e39660b	7294	.	.	425	2406	M		.
8	19810520	01adaa2d0a09b8e6781ebc0bcbc23d4d	7061	.	.	100	221	F		.
9	19810520	01adaa2d0a09b8e6781ebc0bcbc23d4d	5234	5210	.	0	1430	F		.
10	19810520	01adaa2d0a09b8e6781ebc0bcbc23d4d	250	.	.	100	241	F	250	1
11	19680724	025ce9868ffe69b3ea57018f64401c77	4650	4659	250.187	73	334	F	250	1

collapse (sum) DMcount, by(id id_birthday)

	id_birthday	id	acode_icd9_1	acode_icd9_2	acode_icd9_3	drug_amt	t_amt	id_sex	dm	DMcount
1	19820900	012e61103b1db1b6e92ecf423de53aa5	5819	.	.	0	1202	M		.
2	19820921	012e61103b1db1b6e92ecf423de53aa5	250.11	.	.	0	1430	M	250	1
3	19820921	012e61103b1db1b6e92ecf423de53aa5	581	.	.	0	2290	M		.
4	19820921	012e61103b1db1b6e92ecf423de53aa5	250			0	2000	M	250	1
5	19820921	012e61103b1db1b6e92ecf423de53aa5	4658	250123	.	160	415	M	250	1
6	19740327	014fd9cb0edfc5b57255ee4a9e39660b	7804	7870	.	84	531	M		.
7	19740327	014fd9cb0edfc5b57255ee4a9e39660b	7294	.	.	425	2406	M		.
8	19810520	01adaa2d0a09b8e6781ebc0bcbc23d4d	7061	.	.	100	221	F		.
9	19810520	01adaa2d0a09b8e6781ebc0bcbc23d4d	5234	5210	.	0	1430	F		.
10	19810520	01adaa2d0a09b8e6781ebc0bcbc23d4d	250	.	.	100	241	F	250	1
11	19680724	025ce9868ffe69b3ea57018f64401c77	4650	4659	250.187	73	334	F	250	1

	id_birthday	id	DMcount
1	19820900	012e61103b1db1b6e92ecf423de53aa5	0
2	19820921	012e61103b1db1b6e92ecf423de53aa5	3
3	19740327	014fd9cb0edfc5b57255ee4a9e39660b	0
4	19810520	01adaa2d0a09b8e6781ebc0bcbc23d4d	1
5	19680724	025ce9868ffe69b3ea57018f64401c77	1

Collapse → Do them all together

- gen count=1
- gen dm=regexs(0) if (regexm(acode_icd9_1, "[2][5][0]"))
- replace dm=regexs(0) if (regexm(acode_icd9_2, "[2][5][0]"))
- replace dm=regexs(0) if (regexm(acode_icd9_3, "[2][5][0]"))
- gen DMcount=1 if dm~=""

- collpase (sum) count DMcount (mean) t_amt drug_amt, by(id id_birthday)

	id_birthday	id	count	DMcount	t_amt	drug_amt
1	19820900	012e61103b1db1b6e92ecf423de53aa5	1	0	1202	0
2	19820921	012e61103b1db1b6e92ecf423de53aa5	4	3	1533.75	40
3	19740327	014fd9cb0edfc5b57255ee4a9e39660b	2	0	1468.5	254.5
4	19810520	01adaa2d0a09b8e6781ebc0bcbc23d4d	3	1	630.667	66.6667
5	19680724	025ce9868ffe69b3ea57018f64401c77	1	1	334	73

Drop duplicate observations

- E.g. when you have duplicated id in the NHI ID file.
- duplicate drop id, force

- For example, you want to keep only the most recent insurance in-date for each person:

- gsort id id_birthday –id_in_date
- duplicates drop id id_birthday, force

- To detect duplicate id:
- duplicates list id

	id
1	a
2	b
3	c
4	a
5	a

```
. duplicates list id

Duplicates in terms of id

     obs:   id
     ------------
       1    a
       4    a
       5    a
     ------------
```

Keep the last observation in dd file (alternatively)

- sort id id_birthday in_date out_date
- by id id_birthday in_date: gen a=_n
- by id id_birthday in_date: gen b=_N

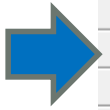
	id	id_birthday	card_seq_no	func_type	in_date	out_date	a	b
1	411e27ec301fb7026d3773f91f79aabc	19451229	E3	AE	20000726	20000801	1	1
2	411e27ec301fb7026d3773f91f79aabc	19451231	E3	AE	20000726	20000801	1	1
3	41c369c19a48b31fe8277ce3e64b5eb5	19981230	A6	10	20000814	20000815	1	1
4	437a1100370a66df83b504758fcd8877	19440627	E4	AA	20000724	20000727	1	1
5	4b4831361f3b28de11f05bdbcb41ec5c	19370105	B3	02	20001104	20001106	1	1
6	4fab28b2972029bc84732fb6edfca7da	19631011	A1	13	19991231	.	1	6
7	4fab28b2972029bc84732fb6edfca7da	19631011	A1	13	19991231	.	2	6
8	4fab28b2972029bc84732fb6edfca7da	19631011	A1	13	19991231	.	3	6
9	4fab28b2972029bc84732fb6edfca7da	19631011	A1	13	19991231	.	4	6
10	4fab28b2972029bc84732fb6edfca7da	19631011	A1	13	19991231	.	5	6
11	4fab28b2972029bc84732fb6edfca7da	19631011	A1	13	19991231	20001130	6	6
12	53b0703b0ec68d9650d3038d463a5cd3	19420921	I5	AE	20000912	20000915	1	1
13	53b0703b0ec68d9650d3038d463a5cd3	19420921	I5	AE	20001120		1	3
14	53b0703b0ec68d9650d3038d463a5cd3	19420921	I5	AE	20001120	.	2	3
15	53b0703b0ec68d9650d3038d463a5cd3	19420921	I5	AE	20001120	.	3	3

- keep if a ==b

Reshape: wide to long

- When you have separate waves in different data files and you want to run longitudinal analysis.

	id	income2001	income2000	income2002	sex
1	a	1000	1234	5000	M
2	b	2080	3213	4534	F
3	c	3213	999	2345	M
4	d	2343	1234	33	F
5	e	1923	2323	234	F



	id	year	income	sex
1	a	2000	1234	M
2	a	2001	1000	M
3	a	2002	5000	M
4	b	2000	3213	F
5	b	2001	2080	F
6	b	2002	4534	F
7	c	2000	999	M
8	c	2001	3213	M
9	c	2002	2345	M
10	d	2000	1234	F
11	d	2001	2343	F
12	d	2002	33	F
13	e	2000	2323	F
14	e	2001	1923	F

reshape long income, i(id) j(year)

```
. reshape long income, i(id) j(year)
(note: j = 2000 2001 2002)

Data                               wide   ->   long
-----------------------------------------------------------------------------
Number of obs.                        5   ->       15
Number of variables                   5   ->        4
j variable (3 values)                     ->   year
xij variables:
       income2000 income2001 income2002   ->   income
-----------------------------------------------------------------------------
```

- **reshape long income, i(id id_birthday) j(year)**
- "year" is the name of the time variable that you want STATA to generate. You can call it anything (e.g: month, wave etc).
- If id is not unique:

```
. reshape long income, i(id) j(year)
(note: j = 2000 2001 2002)
i=id does not uniquely identify the observations;
there are multiple observations with the same value of id.
Type "reshape error" for a listing of the problem observations.
r(9);
```

Reshape: long to wide

	id	year	income	sex
1	a	2000	1234	M
2	a	2001	1000	M
3	a	2002	5000	M
4	b	2000	3213	F
5	b	2001	2080	F
6	b	2002	4534	F
7	c	2000	999	M
8	c	2001	3213	M
9	c	2002	2345	M
10	d	2000	1234	F
11	d	2001	2343	F
12	d	2002	33	F
13	e	2000	2323	F
14	e	2001	1923	F

	id	income2000	income2001	income2002	sex
1	a	3213	2080	4534	F
2	b	1234	1000	5000	M
3	c	999	3213	2345	M
4	d	1234	2343	33	F
5	e	2323	1923	234	F

- reshape wide income, i(id) j(year)

```
. reshape wide income, i(id) j(year)
(note: j = 2000 2001 2002)

Data                               long   ->   wide
-----------------------------------------------------------------------------
Number of obs.                       15   ->      5
Number of variables                   4   ->      5
j variable (3 values)              year   ->   (dropped)
xij variables:
                                 income   ->   income2000 income2001 income2002
-----------------------------------------------------------------------------
```

Loops in STATA

- Use loops wisely can save you a lot of time.
- We will demonstrate many cases where loops may be appropriate.

- foreach

Example 1

famid	inc1	inc2	inc3	inc4	inc5	inc6	inc7	inc8	inc9	inc10	inc11	inc12
1	3281	3413	3114	2500	2700	3500	3114	3319	3514	1282	2434	2818
2	4042	3084	3108	3150	3800	3100	1531	2914	3819	4124	4274	4471
3	6015	6123	6113	6100	6100	6200	6186	6132	3123	4231	6039	6215

- Suppose you want to compute the amount of tax (10%) paid for each month, you can do this:

```
generate taxinc1 = inc1 * .10
generate taxinc2 = inc2 * .10
generate taxinc3 = inc3 * .10
generate taxinc4 = inc4 * .10
generate taxinc5 = inc5 * .10
generate taxinc6 = inc6 * .10
generate taxinc7 = inc7 * .10
generate taxinc8 = inc8 * .10
generate taxinc9 = inc9 * .10
generate taxinc10= inc10 * .10
generate taxinc11= inc11 * .10
generate taxinc12= inc12 * .10
```

- But what if you have a thousand variables to do? For example, 50 years of data.

Example 1 (con't)

```
foreach var of varlist inc1-inc12 {
  generate tax`var' = `var' * .10
}
```

- The first time we cycle through the statements, the value of **var** will be **inc1** and the second time the value of **var** will be **inc2** and so on.
- **`var'** . The ` is the quote right below the ~ on keyborad and the ' is the quote below the " on keyboard.
- The first time through the loop, **`var'** is replaced with **inc1**, so the statement:
- **generate tax`var' = `var' * .10** becomes
- **generate taxinc1 = inc1 * .10**

Example 2

- Assume the hypothetical income data below, and you want to generate a variable containing the cumulated income (e.g. when calculating Gini coefficient).

	id	income
1	a	200
2	b	135
3	c	1876
4	d	999
5	e	435
6	f	1256
7	g	3257
8	h	321

sort income

	id	income
1	b	135
2	a	200
3	h	321
4	e	435
5	d	999
6	f	1256
7	c	1876
8	g	3257

	id	income	cumulated
1	b	135	135
2	a	200	335
3	h	321	656
4	e	435	1091
5	d	999	2090
6	f	1256	3346
7	c	1876	5222
8	g	3257	8479

```
gen cumulated = income[1] in 1
local i=2
while `i' < 9 {
replace cumulated = income[`i'] + cumulated[`i'-1] in `i'
local i = `i'+1
}
```

- Very useful in analyzing NHI data with many years.
- For example, I want to create a variable determining each subject's income group for 2000~2009:

- local j=0
- while `j'<=9
- use "my ID 200`j'", clear
- gen income=1 if ins_amt<=40000
- replace income =2 if ins_amt>40000
- save "my ID 200`j'", replace
- Local j=`j'+1

Commands/Syntax covered

- findit
- set mem, set maxvar, set matsize
- compress
- clear
- use
- log using
- inputst
- save
- #delimit;
- label
- List
- rename
- tabstat
- tab

- keep
- drop
- generate (or gen)
- replace
- sort and by
- merge
- append
- Date (DMY etc)
- duplicates drop
- regexs, regexm
- encode, decode
- destring, tostring
- Collapse
- Reshape
- forach, local

In class exercise

- Use the SAS NHI *id* and *cd* files provided, do the following:
- 1. Transfer the two SAS files into STATA format.
- 2. Generate 3 new variables in **id_2005** containing
- (1) How many people did not have any claim in 2005 (ans:478).
  - (2) mean number of outpatient visited for those who had a claim (ans:9.39).
  - (3) 2005 average outpatient costs by person (for those who had at least one utilization), just use t_amt. (calculate one mean) (ans:837.5)
  - (4) How many people had a “250” diagnose (including all extensions) in any of the three diagnosis (acode_icd9_1, acode_icd9_2, acode_icd9_3)? (ans:210)
  - **You must show me your do-file, and the last command should be summarizing the three new variables you created)**
- **學統計軟體就只有持續的練習，沒有捷徑!**

STATA CRASH COURSE

STATISTICAL TESTS AND REGRESSIONS (檢定與迴歸分析)

t-test

- **ttest mpg==25** (one-sample mean-comparison test)

```
. ttest mpg==25

One-sample t test

Variable |     Obs        Mean    Std. Err.   Std. Dev.   [95% Conf. Interval]
---------+--------------------------------------------------------------------
     mpg |      74     21.2973    .6725511    5.785503    19.9569    22.63769
------------------------------------------------------------------------------
    mean = mean(mpg)                                              t =  -5.5055
Ho: mean = 25                                     degrees of freedom =       73

    Ha: mean < 25               Ha: mean != 25                 Ha: mean > 25
 Pr(T < t) = 0.0000         Pr(|T| > |t|) = 0.0000          Pr(T > t) = 1.0000
```

t-test

- **ttest mpg1==mpg2** (two-sample mean-comparison test, paired)

```
. ttest mpg1==mpg2

Paired t test
```

Variable	Obs	Mean	Std. Err.	Std. Dev.	[95% Conf.	Interval]
mpg1	12	21	.7881701	2.730301	19.26525	22.73475
mpg2	12	22.75	.9384465	3.250874	20.68449	24.81551
diff	12	-1.75	.7797144	2.70101	-3.46614	-.0338602

```
    mean(diff) = mean(mpg1 - mpg2)                                t =  -2.2444
Ho: mean(diff) = 0                             degrees of freedom =       11

Ha: mean(diff) < 0           Ha: mean(diff) != 0           Ha: mean(diff) > 0
Pr(T < t) = 0.0232         Pr(|T| > |t|) = 0.0463          Pr(T > t) = 0.9768
```

- Or:
- ttest mpg1 == mpg2, unpaired (for unpaired data)

t-test

- **ttest mpg, by(treated)** (two-group mean-comparison test

	mpg	treated
1	20	0
2	23	0
3	21	0
4	25	0
5	18	0
6	17	0
7	18	0
8	24	0
9	20	0
10	24	0
11	23	0
12	19	0
13	24	1
14	25	1
15	21	1
16	22	1
17	23	1
18	18	1
19	17	1
20	28	1
21	24	1
22	27	1
23	21	1
24	23	1

```
. ttest mpg, by(treated)

Two-sample t test with equal variances

   Group |     Obs        Mean    Std. Err.   Std. Dev.   [95% Conf. Interval]
---------+--------------------------------------------------------------------
       0 |      12          21    .7881701    2.730301    19.26525    22.73475
       1 |      12       22.75    .9384465    3.250874    20.68449    24.81551
---------+--------------------------------------------------------------------
combined |      24      21.875    .6264476    3.068954    20.57909    23.17091
---------+--------------------------------------------------------------------
    diff |           -1.75    1.225518               -4.291568    .7915684
------------------------------------------------------------------------------
    diff = mean(0) - mean(1)                                      t =  -1.4280
Ho: diff = 0                     degrees of freedom =       22

    Ha: diff < 0                 Ha: diff != 0                 Ha: diff > 0
 Pr(T < t) = 0.0837         Pr(|T| > |t|) = 0.1673          Pr(T > t) = 0.9163
```

You can always use the pull-down menu, but habit: always copy the command to your do-files.

Chi-square test

- **tabulate region agecat, chi2**

```
. tabulate region agecat, chi2

   Census |                agecat
   Region |     19-29       30-34       35+ |     Total
----------+---------------------------------+----------
       NE |        46          83        37 |       166
  N Cntrl |       162          92        30 |       284
    South |       139          68        43 |       250
     West |       160          73        23 |       256
----------+---------------------------------+----------
    Total |       507         316       133 |       956

          Pearson chi2(6) =   61.2877   Pr = 0.000
```

- **tabulate region agecat, exact**

ANOVA

- **anova write prog** (one-way)

	prog	write
1	general	52
2	vocation	59
3	general	33
4	vocation	44
5	academic	52
6	academic	52
7	general	59
8	academic	46
9	general	57
10	academic	55
11	vocation	46
12	academic	65
13	academic	60
14	academic	63
15	academic	57
16	general	49
17	academic	52
18	general	57
19	academic	65
20	general	39

```
. anova write prog

                           Number of obs =       200     R-squared     =  0.1776
                           Root MSE      =  8.63918     Adj R-squared =  0.1693

                  Source | Partial SS    df       MS           F     Prob > F
-------------------------+----------------------------------------------------
                   Model |  3175.69786     2  1587.84893       21.27     0.0000
                         |
                    prog |  3175.69786     2  1587.84893       21.27     0.0000
                         |
                Residual |  14703.1771   197   74.635417
-------------------------+----------------------------------------------------
                   Total |   17878.875   199   89.843593
```

- **anova write prog sex sex*prog** **(two-way)**

Frequently used regressions

- OLS (reg)
- Logistic (logit)
- Probit (probit)
- Ordered logit (ologit)
- Ordered probit (oprobit)
- Multinomial logit (mlogit)
- Poisson regression (poisson)
- Negative binomial regression (nbreg)
- Seemingly unrelated regression (sureg)

Linear regression (OLS)

- regress y x1 x2
- reg mpg weight length
- (Y=mpg, X_1= weight, X_2=length)
- (results can simply be copied and pasted to Excel)

```
. reg mpg weight length

      Source |       SS       df       MS              Number of obs =      74
-------------+------------------------------           F(  2,    71) =   69.34
       Model |  1616.08062     2  808.040312            Prob > F      =  0.0000
    Residual |  827.378835    71   11.653223           R-squared     =  0.6614
-------------+------------------------------           Adj R-squared =  0.6519
       Total |  2443.45946    73  33.4720474           Root MSE      =  3.4137

------------------------------------------------------------------------------
         mpg |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
      weight |  -.0038515   .001586    -2.43   0.018    -.0070138   -.0006891
      length |  -.0795935   .0553577   -1.44   0.155    -.1899736    .0307867
       _cons |   47.88487   6.08787     7.87   0.000       35.746    60.02374
------------------------------------------------------------------------------
```

Creating dummy variables (xi:)

```
. tab own
```

own	Freq.	Percent	Cum.
Christy	16	21.62	21.62
Hu	21	28.38	50.00
Jack	22	29.73	79.73
Nicole	11	14.86	94.59
var14	4	5.41	100.00
Total	74	100.00	

	own	_Iown_2	_Iown_3	_Iown_4	_Iown_5
1	Christy	0	0	0	0
2	Jack	0	1	0	0
3	Jack	0	1	0	0
4	Nicole	0	0	1	0
5	Hu	1	0	0	0
6	Hu	1	0	0	0
7	Jack	0	1	0	0
8	Christy	0	0	0	0
9	Nicole	0	0	1	0
10	Christy	0	0	0	0
11	Hu	1	0	0	0
12	Hu	1	0	0	0
13	Jack	0	1	0	0
14	var14	0	0	0	1
15	Christy	0	0	0	0
16	Jack	0	1	0	0
17	Jack	0	1	0	0
18	Nicole	0	0	1	0
19	Hu	1	0	0	0
20	Hu	1	0	0	0
21	Jack	0	1	0	0
22	Christy	0	0	0	0

- xi: regress mpg weight i.own
- (STATA 15 does not require you to type xi)

```
. xi: regress mpg weight  i.own
i.own             _Iown_1-5           (_Iown_1 for own==Christy omitted)

      Source |       SS       df       MS              Number of obs =      74
-------------+------------------------------           F(  5,    68) =   28.06
       Model |  1645.73486     5  329.146973            Prob > F      =  0.0000
    Residual |  797.724596    68  11.7312441            R-squared     =  0.6735
-------------+------------------------------           Adj R-squared =  0.6495
       Total |  2443.45946    73  33.4720474            Root MSE      =  3.4251

------------------------------------------------------------------------------
         mpg |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
      weight |  -.0059294   .0005201   -11.40   0.000    -.0069672   -.0048916
      _Iown_2 |  -1.231024    1.14298    -1.08   0.285    -3.511804    1.049757
      _Iown_3 |    .308643    1.13061     0.27   0.786    -1.947454     2.56474
      _Iown_4 |   -1.69187   1.348067    -1.26   0.214    -4.381895    .9981548
      _Iown_5 |   .9945885   1.914804     0.52   0.605    -2.826343     4.81552
        _cons |   39.65625   1.712349    23.16   0.000     36.23931     43.07319
------------------------------------------------------------------------------
```

Post estimation for regress

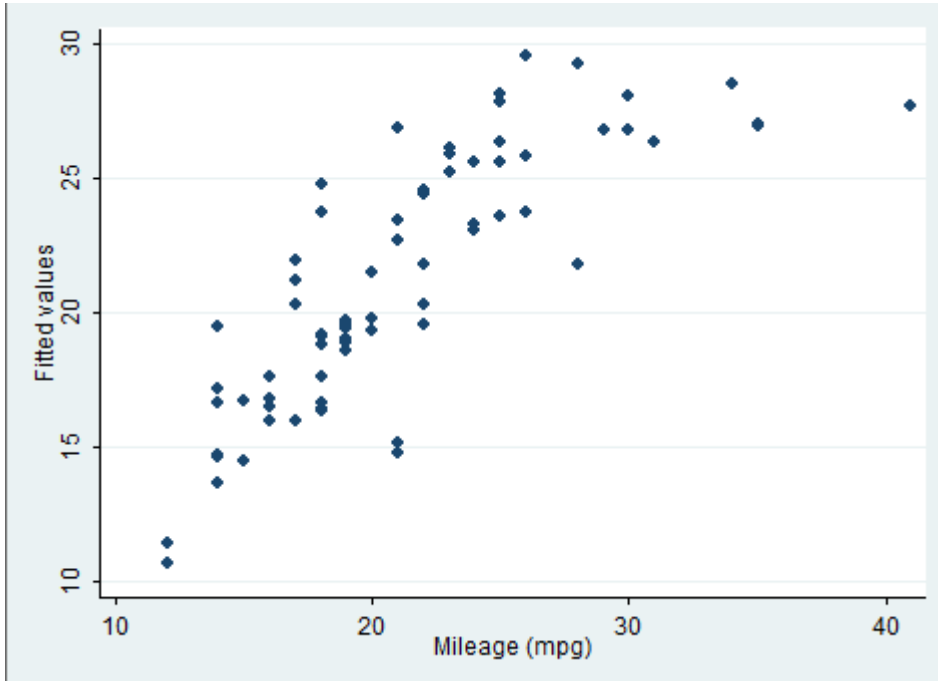
```
. regress mpg weight length

      Source |       SS       df       MS              Number of obs =      74
-------------+------------------------------           F(  2,    71) =   69.34
       Model |  1616.08062     2  808.040312            Prob > F      =  0.0000
    Residual |  827.378835    71   11.653223            R-squared     =  0.6614
-------------+------------------------------           Adj R-squared =  0.6519
       Total |  2443.45946    73  33.4720474            Root MSE      =  3.4137

------------------------------------------------------------------------------
         mpg |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
      weight |  -.0038515   .001586    -2.43   0.018    -.0070138   -.0006891
      length |  -.0795935   .0553577   -1.44   0.155    -.1899736    .0307867
       _cons |   47.88487   6.08787     7.87   0.000       35.746    60.02374
------------------------------------------------------------------------------

. predict yhat
(option xb assumed; fitted values)
```

- twoway (scatter yhat mpg)



For graphics, simply use the pull-down menu and then copy the commands.

- predict resid, residual → this predicts the residual of the estimation.

Logistic regressions

- logit y x_1 x_2 → this gives you the raw estimates

```
. logit foreign mpg weight

Iteration 0:   log likelihood =  -45.03321
Iteration 1:   log likelihood = -29.898968
Iteration 2:   log likelihood = -27.495771
Iteration 3:   log likelihood = -27.184006
Iteration 4:   log likelihood = -27.175166
Iteration 5:   log likelihood = -27.175156

Logistic regression                             Number of obs     =         74
                                                LR chi2(2)        =      35.72
                                                Prob > chi2       =     0.0000
Log likelihood = -27.175156                     Pseudo R2         =     0.3966

------------------------------------------------------------------------------
     foreign |      Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
         mpg |  -.1685869   .0919174    -1.83   0.067    -.3487418    .011568
      weight |  -.0039067   .0010116    -3.86   0.000    -.0058894   -.001924
       _cons |   13.70837   4.518707     3.03   0.002     4.851864   22.56487
------------------------------------------------------------------------------
```

Logistic regressions

- logistic y x1 x2 → this gives you the odds ratios.

```
. logistic foreign mpg weight

Logistic regression                             Number of obs   =         74
                                                LR chi2(2)      =      35.72
                                                Prob > chi2     =     0.0000
Log likelihood = -27.175156                     Pseudo R2       =     0.3966

------------------------------------------------------------------------------
     foreign | Odds Ratio   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
         mpg |   .8448578   .0776572    -1.83   0.067     .7055753    1.011635
      weight |   .9961009   .0010077    -3.86   0.000     .9941279    .9980779
------------------------------------------------------------------------------
```

- predict phat →the default, calculates the probability of a positive outcome.
- xb calculates the linear prediction.
- stdp calculates the standard error of the linear prediction.

Interaction terms

- reg Y x##z (full specification)
- reg Y c.x##z
- reg Y x z x#z (same as the full specification)
- reg Y x x#z (simple effect, same model, different representation)
- reg Y z x#z (simple effect, same model, different representation)

Example

- COCI=0,1,2,3
- AGE=1,2,3

```
. reg ERQ i.COCI_G_##i.AGE_G_

      Source |       SS       df       MS              Number of obs =   30565
-------------+------------------------------           F( 11, 30553) =   15.40
       Model |  435.840419    11  39.6218563           Prob > F      =  0.0000
    Residual |  78607.9916 30553  2.57284036           R-squared     =  0.0055
-------------+------------------------------           Adj R-squared =  0.0052
       Total |  79043.832 30564  2.58617432           Root MSE      =   1.604

------------------------------------------------------------------------------
         ERQ |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
      COCI_G_ |
          1  |   .3553899   .1043324     3.41   0.001      .150894    .5598857
          2  |    .270629   .1070403     2.53   0.011     .0608255    .4804325
          3  |   -.025002    .077158    -0.32   0.746    -.1762349     .126231
             |
      AGE_G_ |
          2  |   .1282751    .055023     2.33   0.020     .0204277    .2361226
          3  |   .1836687   .0568497     3.23   0.001      .072241    .2950965
             |
COCI_G_#AGE_G_ |
        1 2  |  -.2577499   .1109142    -2.32   0.020    -.4751462   -.0403535
        1 3  |  -.1001503   .1167473    -0.86   0.391    -.3289799    .1286793
        2 2  |  -.3089093   .1135565    -2.72   0.007    -.5314847   -.0863338
        2 3  |  -.1178965   .1184281    -1.00   0.319    -.3500205    .1142275
        3 2  |  -.1259063   .0821819    -1.53   0.126    -.2869862    .0351736
        3 3  |  -.0355818    .085717    -0.42   0.678    -.2035907     .132427
             |
       _cons |   .3155779   .0508505     6.21   0.000     .2159089    .4152469
------------------------------------------------------------------------------
```

```
. reg ERQ i.COCI_G i.COCI_G_#i.AGE_G_
```

Source	SS	df	MS		
Model	435.840419	11	39.6218563	Number of obs =	30565
Residual	78607.9916	30553	2.57284036	F(11, 30553) =	15.40
				Prob > F =	0.0000
				R-squared =	0.0055
				Adj R-squared =	0.0052
Total	79043.832	30564	2.58617432	Root MSE =	1.604

ERQ	Coef.	Std. Err.	t	P>|t|	[95% Conf.	Interval]
COCI_G_						
1	.3553899	.1043324	3.41	0.001	.150894	.5598857
2	.270629	.1070403	2.53	0.011	.0608255	.4804325
3	-.025002	.077158	-0.32	0.746	-.1762349	.126231
COCI_G_#AGE_G_						
0 2	.1282751	.055023	2.33	0.020	.0204277	.2361226
0 3	.1836687	.0568497	3.23	0.001	.072241	.2950965
1 2	-.1294748	.0963038	-1.34	0.179	-.3182341	.0592846
1 3	.0835184	.1019708	0.82	0.413	-.1163487	.2833855
2 2	-.1806341	.0993355	-1.82	0.069	-.3753359	.0140676
2 3	.0657722	.1038909	0.63	0.527	-.1378584	.2694028
3 2	.0023689	.0610436	0.04	0.969	-.1172792	.1220169
3 3	.1480869	.0641522	2.31	0.021	.0223458	.2738279
_cons	.3155779	.0508505	6.21	0.000	.2159089	.4152469

```
. reg ERQ i.AGE_G_ i.COCI_G_#i.AGE_G_
```

Source	SS	df	MS
Model	435.840419	11	39.6218563
Residual	78607.9916	30553	2.57284036
Total	79043.832	30564	2.58617432

Number of obs	= 30565
F(11, 30553)	= 15.40
Prob > F	= 0.0000
R-squared	= 0.0055
Adj R-squared	= 0.0052
Root MSE	= 1.604

ERQ	Coef.	Std. Err.	t	P>|t|	[95% Conf.	Interval]
AGE_G_						
2	.1282751	.055023	2.33	0.020	.0204277	.2361226
3	.1836687	.0568497	3.23	0.001	.072241	.2950965
COCI_G_#AGE_G_						
1 1	.3553899	.1043324	3.41	0.001	.150894	.5598857
1 2	.09764	.037639	2.59	0.009	.0238659	.171414
1 3	.2552395	.0523897	4.87	0.000	.1525535	.3579255
2 1	.270629	.1070403	2.53	0.011	.0608255	.4804325
2 2	-.0382803	.0379137	-1.01	0.313	-.1125926	.0360321
2 3	.1527325	.0506713	3.01	0.003	.0534146	.2520504
3 1	-.025002	.077158	-0.32	0.746	-.1762349	.126231
3 2	-.1509082	.0282931	-5.33	0.000	-.2063639	-.0954526
3 3	-.0605838	.0373368	-1.62	0.105	-.1337655	.0125979
_cons	.3155779	.0508505	6.21	0.000	.2159089	.4152469

Change reference group

- regress Y ib2.edu ib3.source

Generalized estimating equation (GEE)

- GEEs are one of the methods of analysis that account for
- correlated observations.

- • Examples – repeated observations on individuals over
- time, clustered observations (e.g. data grouped by family,
- general practice etc).

- GEEs use robust estimation of standard errors to allow for clustering.

Example

- webuse union
- xtset id year

```
      xtset id year
panel variable:  id (unbalanced)
 time variable:  year, 70 to 88, but with gaps
         delta:  1 unit
```

- To fit a logit GEE:
- xtgee union age grade not_smsa south, family(binomial) link(logit)

```
. xtgee union age grade not_smsa south, family(binomial) link(logit)

Iteration 1: tolerance = .07327489
Iteration 2: tolerance = .00519852
Iteration 3: tolerance = .00024049
Iteration 4: tolerance = .00001086
Iteration 5: tolerance = 4.907e-07

GEE population-averaged model                   Number of obs      =     26200
Group variable:                        id       Number of groups   =      4434
Link:                               logit       Obs per group: min =         1
Family:                          binomial                      avg =       5.9
Correlation:                 exchangeable                      max =        12
                                                Wald chi2(4)       =    229.87
Scale parameter:                        1       Prob > chi2        =    0.0000

------------------------------------------------------------------------------
       union |      Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
         age |   .0098801   .0020824     4.74   0.000     .0057986    .0139616
       grade |   .0606146   .0108383     5.59   0.000     .0393719    .0818573
    not_smsa |  -.1257349   .0483488    -2.60   0.009    -.2204969   -.0309729
       south |  -.5747081    .048645   -11.81   0.000    -.6700506   -.4793656
       _cons |  -2.163394   .1484472   -14.57   0.000    -2.454345   -1.872443
------------------------------------------------------------------------------
```

Example of command: xtgee score age education location [fweight = aaa], family(binomial) link(logit) exposure(treat) corr(unstructured) vce(robust)

Cox proportional hazards model

- Assume the dataset:

	failtime	load	bearings
1	100	15	0
2	140	15	1
3	97	20	0
4	122	20	1
5	84	25	0
6	100	25	1
7	54	30	0
8	52	30	1
9	40	35	0
10	55	35	1
11	22	40	0
12	30	40	1

stset failtime → Declare data to be survival-time data

stcox load bearings → Fit Cox proportional hazards model

```
. stset failtime

     failure event:  (assumed to fail at time=failtime)
obs. time interval:  (0, failtime]
 exit on or before:  failure

------------------------------------------------------------------------------
       12  total obs.
        0  exclusions
------------------------------------------------------------------------------
       12  obs. remaining, representing
       12  failures in single record/single failure data
      896  total analysis time at risk, at risk from t =         0
                            earliest observed entry t =         0
                                 last observed exit t =       140
```

```
Cox regression -- Breslow method for ties

No. of subjects =           12                     Number of obs    =        12
No. of failures =           12
Time at risk    =          896
                                                   LR chi2(2)       =     23.39
Log likelihood  =    -8.577853                     Prob > chi2      =    0.0000

------------------------------------------------------------------------------
          _t |      Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
        load |    .4229578   .1433485     2.95   0.003     .1419999    .7039157
    bearings |   -2.754461   1.173115    -2.35   0.019    -5.053723   -.4551981
------------------------------------------------------------------------------
```

Kaplan-Meier

- webuse drug2b
- sts graph, by(drug)