経済分析のための Stata 入門

経済産業研究所計量分析データ室　松浦寿幸

早稲田大学大学院経済学研究科　佐々木明果

慶應義塾大学大学院経済学研究科　渡辺善次

2006/04/01 version[†]

[†] 最新版は、http://park1.wakwak.com/~mt_tosiyuki/stata-manual.htm からダウンロード可能。

目 次

第１章　Stata のはじめの一歩

１－１．Stata の起動

まずは、Stata を起動してみましょう。インストール後に Stata を起動すると、以下の４つのウインドウが現れます。以下、簡単にそれぞれのウインドウの役割について説明します。

図１－１

（１）　Stata Command：コマンドを入力するウインドウです。

（２）　Review：過去に実行したコマンドが順次表示されていきます。表示されているコマンドをクリックすると、Stata Command ウインドウに表示されます。

（３）　Variable：使用できる変数の一覧が表示されます。

（４）　Stata　Results：データ処理の結果が表示されます。

実際のデータ処理にあたっては、メニューから処理方法を指定したり、Stata Command ウインドウにコマンドを直接入力したりすることで作業を進めることになります。初心者にはメニューから処理方法を指定するほうが簡単ですが、ここでは Command ウインドウへコマンドを入力して作業を進める方法を中心に説明します。この方法で Stata を操作することに慣れておくと、プログラムを利用する際に移行しやすいからです。

１－２．データの読み込み

Stata では、拡張子が.dta となっている Stata 形式ファイルしか処理に用いることはできません。そこで、まず、Stata 形式のファイルを用意する必要があります。しかし、通常、処理の前段階におけるデータは EXCEL 形式やタブ区切り、カンマ区切り（CSV）などで保存されている場合がほとんどですから、ここでは、これらのファイル形式のデータを Stata に読み込む方法を検討しましょう。

ここでは、以下のような上場企業の財務データを読み込む場合を検討します。

表１－２

証券コード	漢字略称	売上高	経常利益	賃金俸給	試験研究費	従業員数
6502	Toshiba	3256247	53741	498829	168295	74558
6503	Mitsubishi	2394085	30059	459219	138355	49842
6504	Fuji	582267	5550	99405	24711	14094
6505	Toyodenki	37643	−1417	9463	650	1344
6506	Yasukawa	124863	123	27822	586	4576
6507	Shinko	88047	−895	20009	845	2571
6508	Medensha	185874	3347	43393	1949	5130

まず、下準備として、変数名を変更します。**Stata は日本語に対応していませんので、変数名が日本語の場合、文字化けしてしまいます。**かならず変数名は半角英数字を用いてください。表１－２のようなデータセットであれば、1行目の日本語変数名は削除してから読み込ませてください。

※変数名に、スペースやハイフンは使えません。”R-and-D”は、”RandD”になってしまいます。どうしても使いたい場合は、アンダーバー（”R_and_D”）を用いましょう。

code	name	sales	Profit	Wage	R_and_D	Labor
6502	Toshiba	3256247	53741	498829	168295	74558
6503	Mitsubishi	2394085	30059	459219	138355	49842
6504	Fuji	582267	5550	99405	24711	14094
6505	Toyodenki	37643	−1417	9463	650	1344
6506	Yasukawa	124863	123	27822	586	4576
6507	Shinko	88047	−895	20009	845	2571
6508	Medensha	185874	3347	43393	1949	5130

１－２－１．タブ区切り・カンマ区切り（CSV）のファイルの読み込み

insheet コマンドを用います。表１－２のデータが CSV ファイル（たとえば、D ドライブの¥Data フォルダー内に profit-loss.csv というファイル名とします。）で保存されていると
すると、以下のようなコマンドを Command ウインドウに書き込みます。

insheet using d:¥Data¥profit-loss.csv

なお、表頭に変数名を入力しておくと、Variable ウインドウに変数名が表示されます。入力されていない場合、変数名は、v1、v2、v3…となります。この場合、rename コマンドで変数名を変更できます。rename の使い方は、

rename [旧変数名] [新変数名]

となります。

また、複数のファイルの読み込みを行う際は、作業用フォルダーを指定することもできます。たとえば、Dドライブの¥Dataフォルダーを作業用フォルダーとすれば、以下のコマンドは、上記の読み込みコマンドと同じ意味になります。

cd d:¥Data
insheet using profit-loss.csv

また、現在指定している作業用フォルダーを確認する際は、pwd コマンドを用います。

. pwd
D:¥Data

１－２－２．エクセルファイルの読み込み

エクセルファイルの読み込み方法はいくつか方法があります。

(1) タブ区切り、カンマ区切り（CSV 形式）で保存しなおして、insheet コマンドで入力する。

(2) コピー＆ペーストで貼り付ける
メニューの「Data」をクリックすると、「Data Editor」が開きます。あらかじめ EXCEL で入力したいデータを範囲指定して「コピー」しておき、「Data Editor」が開いている状態で、メニューの「Edit」→「Paste」とすれば、簡単にデータを読み込ませることができます。

※「Do ファイルについて」で詳述しますが、ここでは(1)の方法をお勧めします。(1)の方法の場合、Do ファイルとして作業をプログラム化させておくことができるので、後になって、最初に EXCEL で作成したデータを補正したり、変数を追加する場合、データの読み込み遡って作業をやり直すことができるからです。

１－３．データを保存する

データを読み込んだら、まず、Stata 形式でデータを保存しましょう。保存の仕方には二通りあって、メニューの「file」→「Save」or「Save as」で保存するか、Command ウインドウで、save コマンドを入力してください。新規ファイルの作成の場合は以下のようになります。

save D:¥Data¥profit-loss.dta

なお、既存のファイルに上書きする場合、“, replace”オプションを付けます。すなわち、

save D:¥Data¥profit-loss.dta ,replace

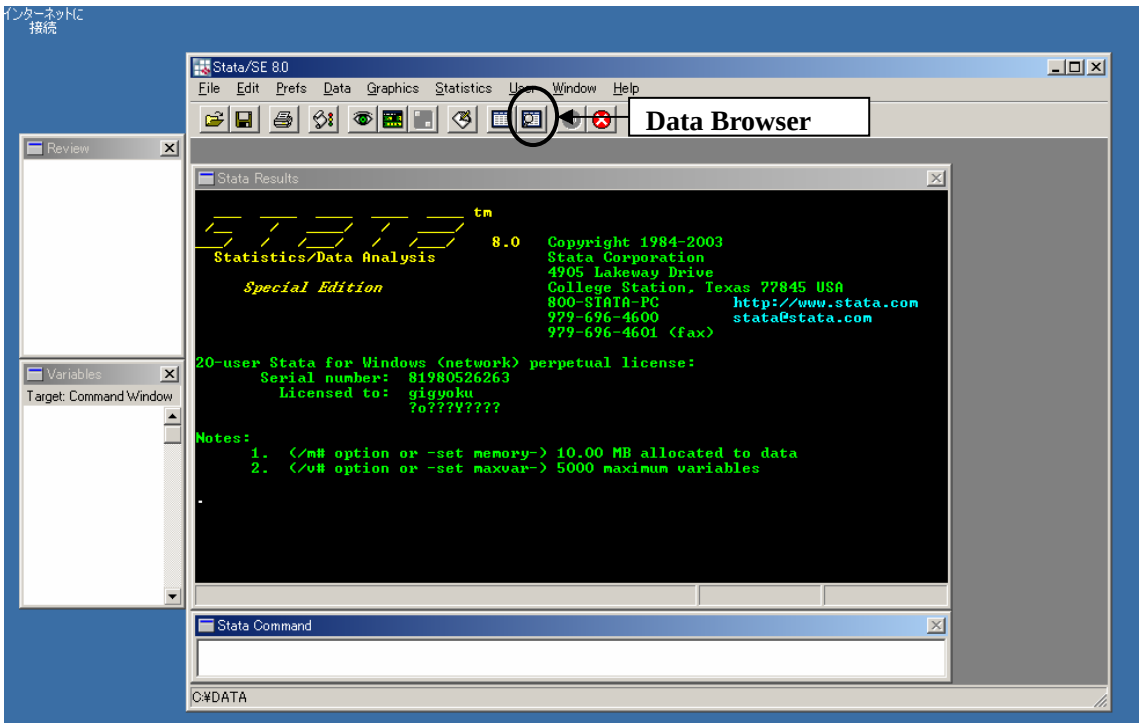
とします。なお、一度、保存したファイルを開くには、use コマンドを用います。

use D:¥Data¥profit-loss.dta

１－４．読み込んだデータを確認しよう

データを保存したら、読み込んだデータを確認しましょう。メニューに、ワークシートの形をしたアイコンが二つあるのがわかるでしょうか？右側が Data Browser です。（図１－４－１）ここをクリックすると、ワークシートが現れるので、データがきちんと入力されているか確認しましょう。なお、Data Browser ではエクセルのように直接データを加工することはできません。

図１－４－１



変数がたくさんある場合、Data Browser では、データは一度に表示されないので、スクロールさせる必要があります。面倒な場合は、必要な変数だけを表示させたり、ある条件を満たすデータだけを表示させたりすることもできます。具体的には、Command ウインドウに、

browse sale profit

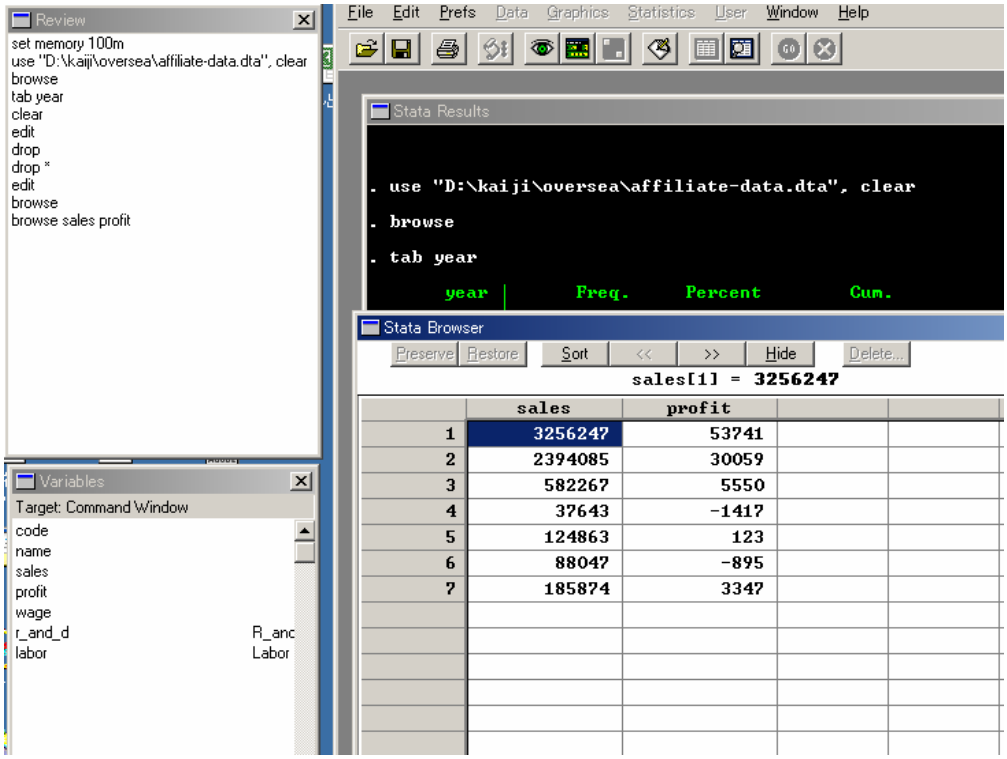
と入力すると、sale と profit だけが表示されます。（図１－４－２）

また、sale が 100,000 以下の企業だけを表示させたいときは、

browse if sale<=100000

とします。条件式、”if” の使い方については後述します。

図１－４－２



このほか、Results ウインドウ上で、データを確認する方法がいくつかあります。

(1) list variable1 variable2 …:変数(variable)の内容を表示

```
list  sales profit wage

     +-----------------------------+
     |   sales   profit     wage |
     |-----------------------------|
  1. | 3256247    53741   498829 |
  2. | 2394085    30059   459219 |
  3. |  582267     5550    99405 |
  4. |   37643    -1417     9463 |
  5. |  124863      123    27822 |
     |-----------------------------|
  6. |   88047     -895    20009 |
  7. |  185874     3347    43393 |
     +-----------------------------+
```

(2) describe:標本数、変数の属性を表示します。(”des”と省略可)

```
. des

Contains data
  obs:             7
 vars:             7
 size:           252 (99.9% of memory free)
-------------------------------------------------------------------------------
              storage  display     value
variable name   type   format      label      variable label
-------------------------------------------------------------------------------
code            int    %8.0g
name            str10  %10s
sales           long   %12.0g
profit          long   %12.0g
wage            long   %12.0g
r_and_d         long   %12.0g                 R_and_D
labor           long   %12.0g                 Labor
-------------------------------------------------------------------------------
Sorted by:
     Note:  dataset has changed since last saved
```

※ ”storage type”は変数の形式です。”int”は、整数、”str10”は 10bite 以下の文字列、long は long 形式であることを示します。

(3) sum variable1 variable2 …:変数(variable)の基本統計量を表示します。

```
. sum  name sales profit

    Variable |       Obs        Mean    Std. Dev.       Min        Max
-------------+--------------------------------------------------------
        name |         0
       sales |         7      952718     1315259       37643    3256247
      profit |         7    12929.71     21080.1       -1417      53741
```

※ ”name”は、文字列ですので、基本統計量が計算されません。

１－５． 変数の加工と条件式

変数を加工する

表を作成する際に、変数を足したり引いたり、掛けたり割ったりという作業が必要となる場合が出てきます。そんなときに使えるコマンドを整理しておきましょう。

（１） generate：新たな変数を作ったり、変数を加工する場合に使用（gen と省略可）

gen newvar1 = variable1 + variable2
gen newvar2 = variable1 - variable2
gen newvar3 = variable1 * variable2
gen newvar4 = variable1 / variable2

（例）generate hosdc2=hosd1 + hosd2
generate age2 = age*age

（２） egen：gen コマンドには使えないいくつかの関数を使うことができるコマンド。

egen newvar = function(variable1)

function のところには、関数を書き込みます。利用できるものを多数ありますが、主なものは以下のとおりです。

mean：平均値
sum：合計
max：最大値
min：最小値

（例）egen avg = mean(chol)
この例では、chol の平均値を計算し、その値を avg に代入する。

※ 注意点
generate の sum( )は、変数を上から順番に合計した値を順次表示していくが、
egen の sum( )は、変数をすべて合計した値が常に表示される。
gen sum1 = sum(A)
egen sum2 = sum(A)

A	Sum1	Sum2
1	1	15
2	3	15
3	6	15
4	10	15
5	15	15

また、パネルデータを作成する際は、次の **group** 関数が便利です。たとえば、以下のように2年分の都道府県データに対して、**group** 関数を使って新しい変数を作成してみましょう。

Prefecture	Year
Hokkaido	1990
Hokkaido	1991
Aomori	1990
Aomori	1991
Iwate	1990

egen newvar = group(year)

newvar	prefecture	year
1	Hokkaido	1990
2	Hokkaido	1991
1	Aomori	1990
2	Aomori	1991
1	Iwate	1990

year で特定されるグループについて、同じ数値が割り当てられます。カッコ内には、カテゴリー変数（上の例では、prefecture）を指定することもできます。また、カッコ内に複数の変数を並べることもできます。

（３） replace：すでに存在する変数の値などを書き換える時に用いる

replace oldvar = value1 if variable==2

variable が 2 の場合、oldvar の値を value1 に置き換える。
（例）： 変数の値を書き換える（－８ → ５）
replace odd = 5 if odd == －8

Odd	Even
1	2
3	4
-8	6
7	8
9	10

⇒

Odd	Even
1	2
3	4
5	6
7	8
9	10

（４） 条件式の書き方

これまで度々登場していますが、ここで条件式の書き方についてまとめておきましょう。

等しいとき（==）
replace newvar=1 if var1==0
等しくない（!= もしくは、~=）
replace newvar=1 if var1!=0
replace newvar=1 if var1~=0
大小関係（>, <, <=, >=）
replace newvar=1 if var2=>0
「かつ」（&）
replace newvar=1 if var2=>0&var1==0
「または」（|）
replace newvar=1 if var2=>0|var1==0
A かつ B、または、C かつ D
replace newvar=1 if (var3==”A”&var4==”B”)| (var3==”C”&var4==”D”)
※変数が文字列であっても” ” で囲むことで、条件式に加えることができます。

１－６．Do ファイルのすすめ

Doファイルについてふれておきましょう。Doファイルはとは、Stataのコマンドを作業工程順に書き並べたファイルで、いくつものコマンドをまとめて実行する際、たいへん便利です。また、作業工程をすべて DO ファイル上で記述する習慣をつけておけば、すべての作業をもう一度初めからやり直すことができます。人間というものは、かならずミスをする動物ですから、作業を繰り返しているうちに、どこかでミスをしてしまうものです。そんな場合も、一連の作業を Do ファイル上で記述しておけば、元に戻ってデータセットを修正することができるわけです。

さて、Do ファイルの作成方法ですが、秀丸などのテキストエディターで、Stata のコマンドを書き込んだファイルを作成して、保存するときに拡張子を“.do”とします。Stataには専用のDoファイルエディターがありますが、使い勝手はあまりよくありません。以下の例は、これまでの一連の作業をDOファイルにしたものです。

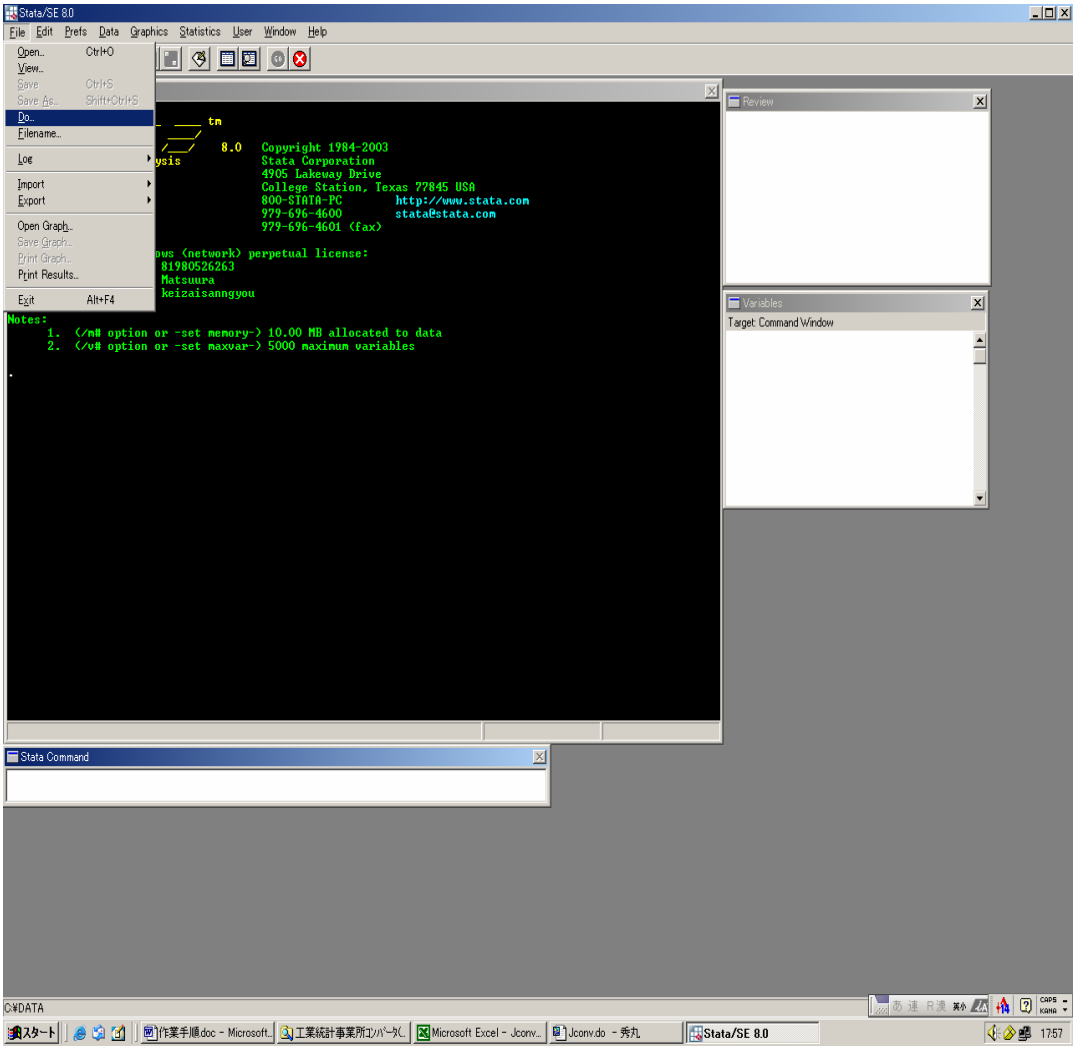
```
* do file の例
insheet using c:¥Data¥profit-loss.csv
des
list sales labor
sum sales labor
```

Do ファイルの中に、コメントを加える場合は、行頭に“*”をつけておきます。また、変数の数がたくさんあって、改行したい場合は、行末に”/*”、次の行の行頭に”*/”を入れます。

```
* Do ファイルで改行したいとき
insheet using c:¥Data¥profit-loss.csv
des
list sales labor wage /*
*/ name profit
```

Doファイルを実行するには、「file」→「do」で、ファイルの所在を指定します。（図１－６参照）

図１－６



Command ウインドウを利用する際は、

do c:¥Data¥yomikomi.do

と入力します。

１－７．LOG をとる

さて、データが正しく Stata に読み込まれたことが確認できたら、いよいよ分析ですが、その前に、ログ（作業記録）のとり方についてみておきましょう。

Stata による作業結果は、Results ウインドウに表示されますが、結果が長くなるとすべてを見ることができなくなります。そこで、Results ウインドウに表示された結果をファイル上に記録する必要が出てくるわけです。

ログファイルを作成するには、メニューの LOG アイコンをクリックします。（図１－７）既存のログ

ファイルを開くことも出来ますが、その際は、結果を既存のファイルに付け足すか（Append を選択）、上書きするかを（Overwrite）を選択します。

Command ウインドウや Do ファイル上で実行したい場合は、

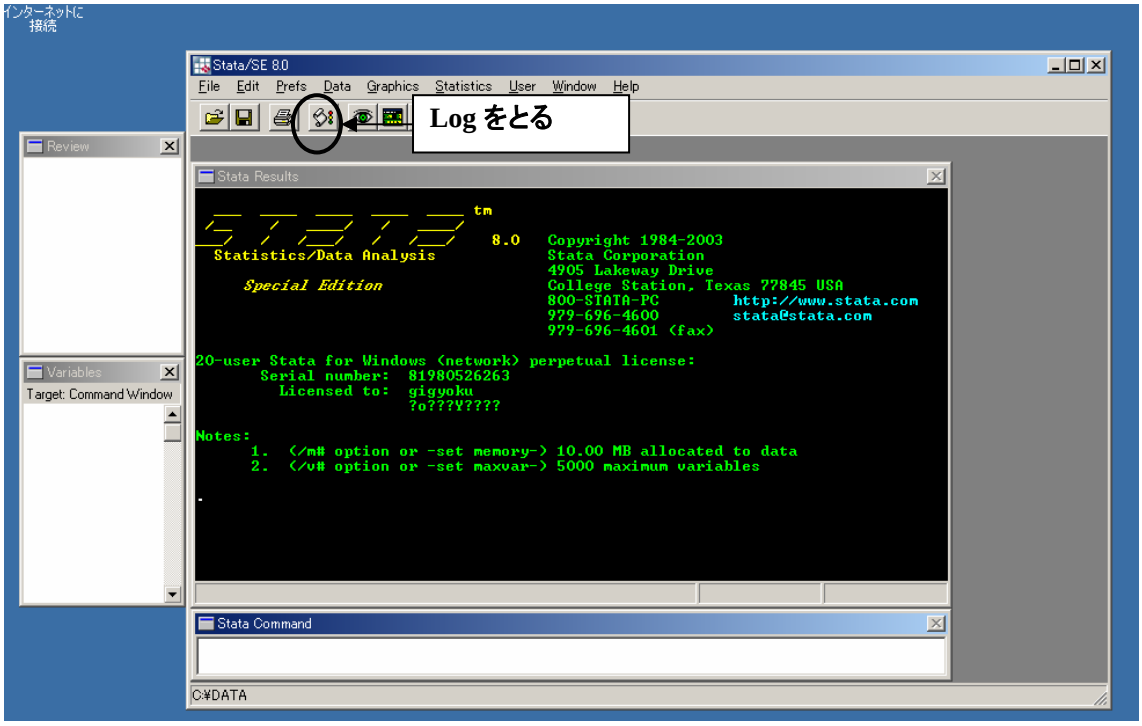
log using D:¥Data¥logwotoru.log

とします。既存の logwotoru.log ファイルにこのコマンド以下の結果を付け足していき、この場合は、

log using D:¥Data¥logwotoru.log, append

上書きする際は、”append”の代わりに”overwrite”と記入します。ここでは、拡張子を”.log”としていますが、必ずしも”.log”である必要はありません。どんな拡張子でもテキストファイトして保存されていますので、秀丸等で開くことが出来ます。

図1－7


インターネットに
接続
Stata/SE 8.0
File Edit Prefs Data Graphics Statistics User Window Help
Log をとる
Review
Variables
Target: Command Window
Stata Results
tm
8.0
Statistics/Data Analysis
Special Edition
Copyright 1984-2003
Stata Corporation
4905 Lakeway Drive
College Station, Texas 77845 USA
800-STATA-PC http://www.stata.com
979-696-4600 stata@stata.com
979-696-4601 (fax)
20-user Stata for Windows (network) perpetual license:
Serial number: 81980526263
Licensed to: gigyoku
?o???Y????
Notes:
1. (/m# option or -set memory-) 10.00 MB allocated to data
2. (/v# option or -set maxvar-) 5000 maximum variables
Stata Command
C:¥DATA

第１章補論

補論１－１．データ読み込みのトラブルシューティング

ここでは、データ読み込みの際のトラブル対処法についていくつか解説しておきます。

（１） メモリーが足りない！

大容量のデータを読み込ませると、以下のようなメッセージが Results ウインドウに表示され、データが読み込めないことがあります。

```
. use "D:¥Data¥daikibo-data.csv", clear
no room to add more observations
    An attempt was made to increase the number of observations beyond what is
currently
    possible.  You have the following alternatives:

     1.  Store your variables more efficiently; see help compress.  (Think of
Stata's
         data area as the area of a rectangle; Stata can trade off width and length.)

     2.  Drop some variables or observations; see help drop.

     3.  Increase the amount of memory allocated to the data area using the set
memory
         command; see help memory.
r(901);
```

このメッセージがでるのは、Stata に割り当てられているメモリーよりもデータのほうが大きいからです。このような場合、データを読み込む前に、Stata に割り当てられるメモリー領域を確保しておく必要があります。たとえば、50m割り振りたい場合は、

set memory 50m

とします。一度、読み込みに失敗し、エラーメッセージが出た後で、メモリーの割当量を変更したい場合は、データセットをクリアーしてください。具体的には、Command ウインドウから以下のように入力します。

clear

（２） browse や list でデータが確認できるのに、sum で記述統計量が出ない！

数値列に文字列が混ざっていると、データ読み込みの際に、その変数は文字列として認識されてしまいます。たとえば、下図のように、欠損値が、”N.A.”と入力されている場合、その変数は文字列となります。

value1	value2	value3
5465	5647	5835
N.A.	5835	6030
4164	4303	4446
4634	N.A.	4446
4355	4500	N.A.

このデータセットを読み込み、descript コマンドで変数の属性を調べると、読み込んだ変数の storage type が str(文字列)になっています。

```
. des

Contains data
  obs:             5
 vars:             3
 size:            52 (99.9% of memory free)
-------------------------------------------------------------------------------
              storage  display     value
variable name   type   format      label      variable label
-------------------------------------------------------------------------------
value1          str4    %8.0g
value2          str4   %10s
value3          str4   %12.0g
-------------------------------------------------------------------------------
Sorted by:
```

このとき、読み込んだ変数について、sumarize で記述統計を出力しようとすると、以下のように結果が出てきません。

```
. sum  name sales profit

    Variable |       Obs        Mean    Std. Dev.       Min        Max
-------------+--------------------------------------------------------
      value1 |         0
      value2 |         0
      value3 |         0
```

このような場合、読み込み前のデータに戻って、**“N.A.”**を、ピリオド”.”に置換するか、空白セルにしてしまいましょう。その後、再度、読み込みを実行してみてください。

また、Stata 上で変換することも出来ます。データの置き換えコマンドある replace を用います。

replace value2=”.” if value2==”N.A.”

このコマンドは、**value2** の要素が、**”N.A.”**になっているものは、ピリオド”.”に置き換えよ、と

いう意味です。ただし、これだけの作業では、まだ変数は文字列のままです。変数の要素がすべて数値、もしくはピリオド”.”になったら、

destring value2,replace

と入力します。
この他、空白セルにゴミ(たとえば、誤って空白セルに、”^”、”;”など)が入っている場合も文字列になってしまいます。このような場合、まずどこにどんなものが入っているのかを探し出すのは大変です。そこで、当該変数で sort variable(variable 内のデータを、大きいもの、もしくはアルファベット順に並び替える)して、その変数を browse してみてください。数値に異物が混入していれば、一番最後に並んでいるはずです。

（３）数値と文字列が組み合わさった変数を分解したい

変数が数値と文字列の組み合わせになっている場合で、それを分解して利用したいケースを考えましょう。

たとえば、以下のような数値と文字が組み合わさった変数があったとします。この変数の上二桁が業種コードで、アルファベットが法人属性(個人企業なら A、法人企業なら B)、下一桁が本店か(1)、支店か(2)を示しているとします。

	code
1	58A1
2	58A2
3	58B1
4	59B2

基本的には、generate コマンドにオプションを付けて処理します。

1)アルファベットを取り出したいとき

gen str1 corp=substr(code,3,1)

新しい変数の属性
この場合、1 byte の文字列

変数 code の 3 文字目から 1 文字取り出す

2)上二桁の数値を取り出したいとき

gen byte industry=real(substr(code,1,2))

新しい変数の属性
この場合、数値

取り出した数値を実数として認識する。
real が無い場合、文字列扱い

結果は、以下のようになります。

	code	corp	industry
1	58A1	A	58
2	58A2	A	58
3	58B1	B	58
4	59B2	B	59

また、この方法を応用すれば、複数のコードを結合させた長い桁数の ID 番号を分解することもできます。
たとえば、以下のような ID 番号があったとします。

	id
1	01201001
2	01201002
3	01301001
4	01304001

ID の上二桁が都道府県番号、次の三桁が市区町村コード、最後の三桁が事業所コードとすると、これを分解する方法を考えましょう。
まず、この変数 id を文字列として認識しなおします。

```
gen str10 code_str=string(id)
```

上二桁を取り出し、prefecture(都道府県)とします。

```
gen byte prefecture=real(substr(code_str,1,2))
```

同様の手順で、市区町村コード、事業所コードを取り出すことができます。

補論１－２．プログラミングによる繰り返し作業

同じような作業を何度も繰り返す必要があるとき、DO ファイルを使ったとしても、いちいち、コマンドを並べるのは面倒です。そんなとき、プログラミングの初歩的な知識があると効率的に作業することができます。

複数の変数に同じ処理を適用したい場合は、**for** を使います。たとえば、以下のデータセットのように、P. 15 ページの表のような欠損値が”**N.A.**”と表示されているデータセットがあったといます。“**N.A.**”を欠損値に変えるには、前述のように

replace value1=”.” if value1==”N.A.”
replace value2=”.” if value2==”N.A.”
replace value3=”.” if value3==”N.A.”

という作業を繰り返す必要があります。この一連の作業を、繰り返しコマンドをつかって処理してみましょう。

for num 1/3: replace valueX=”.” if valueX==”N.A.”

この **for num** コマンドを使うと、Stata は、X のところに順番に1から3の数値を代入し、コマンド処理が 3 回繰り返します。
また、**value1, value2, value3, value4** という4つの変数のそれぞれの比率を計算するときは、

gen ratio12=value1/value2
gen ratio13=value1/value3
gen ratio14=value1/value4
gen ratio23=value2/value3
gen ratio24=value2/value4
gen ratio34=value3/value4

となります。これを**for num**コマンドを使うときは、以下のように**¥**で繰り返す数値を複数定義することもできます。

for num 1/3 ¥ num 2/4: gen ratioXY=valueX/valueY

このコマンドの弱点は、数値を順番に代入するときしか使えない点です。全く異なる名称の複数の変数に対して、繰り返し処理を行う場合は、**foreach** コマンドを使います。

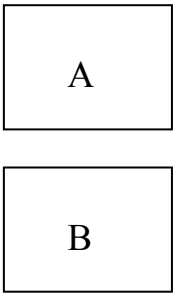
foreach v of varlist income consumption investment [改行]
{ [改行] ①
replace `v'=”.” if `v'==”N.A.” [改行]
} ②

下線部①のところに、処理を施したい変数を並べます。下線部②には、繰り返し処理を施したいコマンドを書きます。このコマンドを実行すると、下線部②の`v’のところに、①の変数が順番に代入されていきます。

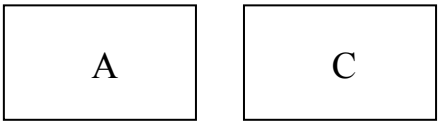
第２章　データベースの作成

第２章では、複数のデータセットをまとめて一つのデータセットにする方法について検討します。データの接続方法としては、A、B、C、D、E をそれぞれ異なるデータセットの入ったファイルとするとき、以下のようなパターンが考えられます。

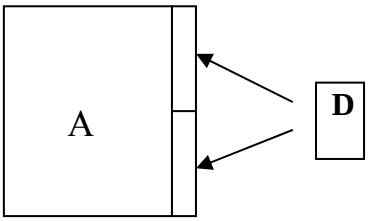
縦に接続する場合　⇒　（２－１）



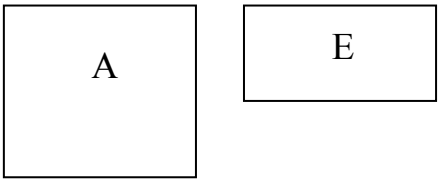
横に接続する場合:完全一致　⇒　（２－２）



片方のデータセットの一部が複数に対応する場合　⇒　（２－３）



横に接続する場合:不完全一致　⇒　（２－４）



以上の（1）～（4）を例をあげながら検討してみましょう。

２－１．データの縦方向の結合（１）

まず、はじめに、複数の個体ごとのデータファイルを結合する方法を考えます。例として、都道府県ごとにファイルされたデータを一つにまとめる方法について考えましょう。

Hokkaido.dta

Prefecture	Year	Production
1	1980	1200
1	1981	1310
1	1982	1450
（省略）		
1	2000	2560

Aomori.dta

Prefecture	Year	Production
2	1980	800
2	1981	710
2	1982	1050
（省略）		
2	2000	1420

この2つのファイルを結合させる場合、append
コマンドを用います。使用方法としては、一方のファイルを開いた状態で、もう一方のファイルを append で呼び出します。
具体的には以下のようになります。（2つのファイルは D:¥Data にあるとします。）

```
cd D:¥Data
use Hokkaido.dta
append Aomori.dta
save Production80-00.dta
```

完成したファイルは以下のようになります。

Prefecture	Year	Production
1	1980	1200
1	1981	1310
（省略）		
1	2000	2560
2	1980	800
2	1981	710
（省略）		
2	2000	1420

append を使う際の注意点として、必ず共通の変数には同じ変数名を付けておいてください。

２－２．データの横方向の結合（２）

次に、複数の個体のデータが変数ごとに各々のファイルに収録されている場合に、データを結合させる例を考えてみましょう。例として、都道府県別の生産額のデータに都道府県別の賃金のデータを接続する方法を考えます。

Wage.dta

prefecture	wage
1	3.616281
2	2.643890
3	3.521620
4	3.630811
5	3.347991
6	3.517322
7	3.928278
8	5.337247
9	4.912243
（省略）	
47	2.687243

Production.dta

prefecture	production
1	18954421
2	4634405
3	4678288
4	8429719
5	3901386
6	4095372
7	7692465
8	11374471
9	7739373
（省略）	
47	3268545

まず、接続する2つのファイルをキーとなる変数で sort しておく必要があります。Wage.dta からみてみましょう。（二つのファイルは、D ドライブの Data フォルダーにあるものとします。）

```
. cd D:¥Data （D ドライブ、Data フォルダーに移動）
. des

Contains data from D:¥Data¥Wage.dta
  obs:            47
 vars:             2                          21 Apr 2004 21:58
 size:           423 (99.9% of memory free)
-------------------------------------------------------------------------------
              storage  display     value
variable name   type   format      label      variable label
-------------------------------------------------------------------------------
prefecture      byte   %8.0g
wage            float  %9.0g
-------------------------------------------------------------------------------
Sorted by:
```

この場合、”Sorted by”の後ろに何も示されていないので、まだ sort されていないことがわかります。そこで、

```
sort prefecture
```

と Command ウインドウに入力し、データをソートしてから、もう一度、des で確認すると、以下のようになります。

```
. des

Contains data from D:¥Data¥Wage.dta
  obs:            47
 vars:             2                          21 Apr 2004 21:58
 size:           423 (99.9% of memory free)
-------------------------------------------------------------------------------
              storage  display     value
variable name   type   format      label      variable label
-------------------------------------------------------------------------------
prefecture            byte   %8.0g
wage                  float  %9.0g
-------------------------------------------------------------------------------
Sorted by:  prefecture
```

この状態で、save しておきます。

save Wage.dta,replace

上書きすることになるので、replace を忘れずに。
同様に、Production.dta も prefecture で sort し、save しておきます。これで準備完了です。
二つのファイルのうち、どちらを先に呼び出して構いませんが、Production.dta を先に呼び出すことにしましょう。

use Production.dta

データを接続するには、merge コマンドを使います。merge コマンドは、

merge [キー変数] using [接続するファイル名]

となります。今の場合、接続のキーとなる変数は prefecture、接続するファイルは Wage.dta ですので、以下のようになります。

merge prefecture using Wage.dta

うまくいけば、データセットは以下のようになります。

prefecture	wage	production	_merge
1	3.616281	18954421	3
2	2.64389	4634405	3
3	3.52162	4678288	3
4	3.630811	8429719	3
5	3.347991	3901386	3
6	3.517322	4095372	3
7	3.928278	7692465	3
8	5.337247	11374471	3
9	4.912243	7739373	3
		（省略）	
47	2.687243	3268545	3

ここで、_merge という新しい変数が生成されていますが、これについては後述します。なお、続けて他のデータセットを merge する場合は、_merge を drop しておいてください。

２－３．少数のデータセットから大規模データに値を割り当てる

さて、２－２のケースでは、接続する2つのファイルの長さは等しくなっていました。しかし、現実のニーズとしては、２－１で作成した都道府県×年次×項目のファイルに、年次別の全国一律のデータ、たとえば物価指数を接続するといった作業が必要になることもあります。このような場合はどうしたらいいのでしょうか？

例として、以下のような年次別の全国平均の物価指数を２－１で作成したデータセットに接続する方法について考えましょう。

Price.dta

Year	Price
1980	100.0
1981	101.2
	（省略）
2000	132.2

接続方法は、基本的に２－２と同じで、まず、接続する際のキーとなる変数で、接続する2つのファイルが sort されているかどうか確認します。この場合は、年次を示す Year がキー変数となります。問題がなければ、一方のデータを開いた状態で、merge を行います。

```
use Production80-00.dta
merge year using Price.dta
```

結果は、うまくいけば、以下の表のようになります。Year が同一のところには、必ず同じ Price の値が入っていることが確認できます。

Prefecture	Year	Production	Price
1	1980	1200	100.0
2	1980	800	100.0
3	1980	1921	100.0
	（省略）		
1	1981	1310	101.2
2	1981	1050	101.2
	（省略）		
1	2000	2560	132.3
2	2000	1420	132.3

２－４．不完全一致のデータセットの接続

（1）～（3）までのデータセットでは、2つのデータセットに含まれるキーとなる変数が完全な対応関係がありました。しかし、実際には、以下のようなキーとなる変数が部分的にしか対応していないケースがままあります。以下のような例を考えましょう。

even.dta

number	Even
5	10
6	12
7	14
8	16

odd.dta

number	Odd
1	1
2	3
3	5
4	7
5	9

この2つのファイルのキーとなる変数は number です。ですが、2つのファイルに重複する変数は、”5”だけです。このケースで、number をキーに merge すると以下のようになります。

use even.dta
merge number using odd.dta

number	Even	Odd	_merge
5	10	9	3
6	12		1
7	14		1
8	16		1
1		1	2
2		3	2
3		5	2
4		7	2

この場合、even.dta と odd.dta の number 変数で共通なのは「５」のみなので、キーとして指定した変数が共通する場合のみ同じ行に odd.dta が接続され、異なる場合には異なる行に odd.dta を接続されます。

なお、merge コマンドを実行すると、_merge という変数が副産物として生成されます。_merge は、二つのデータの結合状態を表します。

_merge=3 : キーに指定した変数が結合前の二つのファイル双方に存在していた場合。
_merge=1 : キーにした変数が、merge 実行前に開いていたファイルのみに存在していた場合。
_merge=2 : キーにした変数が、merge 実行時に呼び出しファイルのみに存在していた場合。

even.dta と odd.dta の接続を例にすると、

even.dta と odd.dta の両方のファイルに含まれていたデータ: _merge=3
even.dta のみに含まれていたデータ: _merge=1
odd.dta のみに含まれていたデータ: _merge=2

となります。

第３章　記述統計による分析（表の作成）

３－１．カテゴリー区分された変数（質的データ）

カテゴリー区分されたデータとは、主に質的(離散)データを指します。一般的には、その区分か数値なのか文字なのかは問われません。大きく分けると①順位尺度と②名義尺度の２種類の尺度により区分されます。順位尺度は、例えば銀行の預金格付けのように、信用度の高い順から AAA～C まで区分されるように、順位を表わす質的データになります。名義尺度は、性別(男、女)や結婚の有無(既婚、未婚、既婚暦有独身、他)などのように特性を表わす質的データです。

以下のデータセット例から STATA による記述統計の取り方をみましょう。(特に記載のない限り３－１～３－２を通じて以下の同一データセット例を使ってコマンド例を紹介することとします。)

３－１、３－２で扱う共通データセット例

id	time	yesno	age	family	y	x1	x2
10001	1	0	36	3	801.2	250.1	22.8
10001	2	0	37	3	840.0	200.5	26.7
10001	3	1	38	4	845.3	287.5	19.4
10002	1	1	24	2	523.0	184.1	35.8
10002	2	1	25	2	534.1	197.8	15.2
10002	3	1	26	2	591.5	205.5	40.9
10003	1	0	31	3	750.0	276.5	55.4
・ ⋮	・ ⋮	・ ⋮	・ ⋮	・ ⋮	・ ⋮	・ ⋮	・ ⋮
11168	2	1	40	4	920.1	321.7	28.9
11168	3	1	41	4	900.4	312.8	52.7

・id と time はデータをパネルデータの構成を表わす変数。
・yesno はカテゴリー区分された変数、その他は連続変数とする。

（１）表の作成

まず、tab コマンドを使ってカテゴリー変数の分布を見ましょう。一変数だけに着目する場合、度数、相対度数、累積相対度数が確認できます。

```
. tab yesno

      yesno |      Freq.     Percent        Cum.
------------+-----------------------------------
          0 |      1,410       40.24       40.24
          1 |      2,094       59.76      100.00
------------+-----------------------------------
      Total |      3,504      100.00
```

ここに条件式(if)を加え、範囲を指定することも可能です。また、plot オプションを加えると、相対度数を視覚的に確認することもできます。

```
. tab yesno if time==1 ,plot

      yesno |      Freq.
------------+-------------+-----------------------------------------------------------
          0 |         480 |**************************************
          1 |         688 |*****************************************************
------------+-------------+-----------------------------------------------------------
      Total |       1,168
```

さらに、二つの変数を指定して分布を確認することもできます。特に指定がない場合、度数のみが表示されます。相対度数を確認するには、行ごとの相対度数(col)、列ごとの相対度数(row)の表示をオプションで指定する必要があります。また、度数表示を省略し、相対度数のみ確認したい時には、nofreq のオプションを指定します。

```
. tab yesno time ,row nofreq

           |                time
     yesno |         1          2          3 |     Total
-----------+---------------------------------+----------
         0 |     34.04      33.83      32.13 |    100.00
         1 |     32.86      33.00      34.15 |    100.00
-----------+---------------------------------+----------
     Total |     33.33      33.33      33.33 |    100.00
```

また、二つの変数を指定する時、all オプションで分布の情報も得られます。

```
. tab yesno time ,all row nofreq

           |                time
     yesno |         1          2          3 |     Total
-----------+---------------------------------+----------
         0 |     34.04      33.83      32.13 |    100.00
         1 |     32.86      33.00      34.15 |    100.00
-----------+---------------------------------+----------
     Total |     33.33      33.33      33.33 |    100.00

          Pearson chi2(2) =   1.5594   Pr = 0.459
 likelihood-ratio chi2(2) =   1.5626   Pr = 0.458
                Cram>'s V =   0.0211
                    gamma =   0.0320  ASE = 0.028
          Kendall's tau-b =   0.0181  ASE = 0.016
```

（２）ラベルの設定

カテゴリー区分された変数には、ラベルを設定することができます。
　銀行の格付けのように複数のカテゴリー(AAA～C)が存在する時、推定のための便宜上、各カテゴリーに数値を与えることがあります(たとえば AAA を 1、AA を 2、…など)。ただしこの時、分布を示す記述統計を取ると、数値に変換されたカテゴリーが示されるため、カテゴリー区分が多ければ多いほど、数値の与え方について混乱してしまいます。ラベルを設定することで、その混乱を回避することができます。以下では、３－１.（1）の **yesno** 変数について例示しましょう。
　下の例の下線部分に適当なラベル名を設定し、続いて①カテゴリー項目、②″ ″ 内に項目ラベル名を指定し、①②を1セットとして必要なカテゴリー項目分のセット数だけ記述します。ただし、必ずしも全カテゴリー項目にラベルを作る必要はありません。

label define yesnolabel 0 "no" 1 "yes"
label value yesno yesnolabel

ここには、指定されたラベルに置き換えられる変数名を指定します。

ラベルを設定したことで、２－１.（1）の表は以下のように表示されます。

yesno	Freq.	Percent	Cum.
no	1,410	40.24	40.24
yes	2,094	59.76	100.00
Total	3,504	100.00	

以下に示すように、ラベルを設定しなかったカテゴリー項目に対し、add オプションを使うことでラベル項目を追加することが可能です。下線部分には追加先の既存ラベル名を指定します。

label define yesnolabel 2 "nuetral" 3 "no answer" , add
label value yesno yesnolabel

また、一度設定したラベルを削除したい場合は、**drop** オプションを使用します。

label drop yesno

複数の変数にラベル設定しているときに、すべてのラベルを一挙に削除したい場合は、変数名を _all とします。すなわち、以下のコマンドを入力します。

label drop _all

３－２．連続変数の記述統計量をみる

（１）sum コマンドによる表示

5種類の基本的な統計量(度数、平均、標準偏差、最小値、最大値)を見る場合は、**sum** コマンドが便利です。条件式を加えることも可能です。

```
. sum y if yesno==1

Variable     |       Obs        Mean    Std. Dev.       Min        Max
-------------+--------------------------------------------------------
           y |      2094    821.1025    490.4918         2       5840
```

通常、統計量は桁数の表示が統一されていません。桁数表示を統一するには **format** コマンドを使います。

format（桁数表示指定をしたい)変数名 %w.df
format コマンドライン中、**w** に表示幅の指定数を、**d** に小数点以下の桁数を記入します。例えば、小数点 2 桁まで表示するとき、以下のように format コマンドの利用により、前ページと桁表示の違いが確認できます。

```
. format y %9.2f
. sum y if yesno==1, format

    Variable |       Obs        Mean    Std. Dev.       Min        Max
-------------+--------------------------------------------------------
           y |      2094      821.10      490.49        2.00    5840.00
```

この例では、変数 y の「全体を 9 桁で、小数点以下を 2 桁の数値で表示せよ」、というコマンドを意味します。

（２）tabstat コマンドによる表示

上の5種類以外の記述統計を見るには **tabstat** コマンドが便利です。具体的には、次の統計量を見ることが可能です。

```
statname    definition
------------------------------------------
mean        mean
count       count of nonmissing observations
n           same as count
sum         sum
max         maximum
min         minimum
range       range = max - min
```

```
sd          standard deviation
var         variance
cv          coefficient of variation (sd/mean)
semean      standard error of mean = sd/sqrt(n)
skewness    skewness
kurtosis    kurtosis
median      median (same as p50)
p1          1st percentile
p5          5th percentile
p10         10th percentile
p25         25th percentile
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"
------------------------------------------------
```

どの統計量を表示するかを **stat( )** の( )内に指定する必要がありますが、この記述がない場合は平均値(mean)のみが表示されます。**tabstat** コマンドでは、得られる統計量が増えるだけでなく、カテゴリー別に記述することも可能となります。以下では、条件式(**if**)を指定し、カテゴリー別(**by**)に表示した例を示しましょう。

```
.  tabstat y if yesno==1 ,by(time) stat(mean n sd sum max min range)

Summary for variables: y
     by categories of: time

    time |      mean         N        sd       sum       max       min     range
---------+--------------------------------------------------------------------------
       1 |  827.9515       688  480.8856  569630.6      5005        44      4961
       2 |  816.2902       691  509.9375  564056.6      5140         3      5137
       3 |   819.163       715   481.002  585701.6      5840         2      5838
---------+--------------------------------------------------------------------------
   Total |  821.1025      2094  490.4918   1719389      5840         2      5838
```

複数の変数に関する記述統計をとることもできます。ここでは、行ごとの各変数が表示されるようcol(variable)と指定しています。col(stat)とすると行ごとに統計量が表示されます。ここでは、表側が表示されませんが、コマンドラインで記述した順番に表示されています。

```
. tabstat age family y x1 x2 ,by(time) stat(mean n sd) col(variable) nototal

Summary statistics: mean, N, sd
  by categories of: time

    time |       age    family         y        x1        x2
---------+--------------------------------------------------
       1 |  32.11387  3.946062  710.2808  229.8031  33.32908
         |      1168      1168      1168      1168      1168
         |  4.314128  1.560986  437.1246  137.7185  68.29036
---------+--------------------------------------------------
       2 |  33.11387   3.97089  701.2497  242.1618  30.23334
         |      1168      1168      1168      1168      1168
         |  4.314128  1.563566  449.4901  150.8053  63.96301
---------+--------------------------------------------------
       3 |  34.11387  4.029966  706.0007  245.3844  31.77663
         |      1168      1168      1168      1168      1168
         |  4.314128  1.539525  431.1006  158.5248  72.25981
------------------------------------------------------------
```

上の例では、コマンドの最後に nototal というオプションがつけてあります。このオプションをつないと、各変数について、全カテゴリー合計の統計量(平均、標本数、標準偏差)も一緒に表示されます。

（３）table コマンドによる表示

table コマンドでは、特に指定がない場合には各データ値に対する度数が表示されるため、３－１.（1）の tab コマンドに類似しています。違いは、table コマンドでは行ごと(col)列ごと(row)の合計値を表示しないという点です。合計値を表示するには row col オプションを加える必要があります。

例） table yesno time, row col
⇒これで“tab yesno time”と同一の表が作成されます。

ただし、tab コマンドは各データ値ごとの度数が表示されるため、連続変数には向かないのに対し、table コマンドでは以下のようにカテゴリー変数に対応した連続変数の統計量を得ることも出来ます。データセット全体の統計量を得るには、row オプションで全データを対象とした統計量を得るのが良いでしょう。また、format( )オプションにより、データの桁表示指定が可能です。( )内の桁数指定方法などは３－２.（1）をご参照ください。

```
. table time, c(mean y sd y mean x1 sd x1) format(%9.2f) row

------------------------------------------------------------
     time |    mean(y)       sd(y)    mean(x1)      sd(x1)
----------+-------------------------------------------------
        1 |     710.28      437.12      229.80      137.72
        2 |     701.25      449.49      242.16      150.81
        3 |     706.00      431.10      245.38      158.52
          |
    Total |     705.84      439.20      239.12      149.37
------------------------------------------------------------
```

カテゴリー変数を指定した後、c( )の( )内に①得たい統計量の種類、②変数名、の①②を1セットとして5セットまで指定できます。①には、以下の統計量を指定可能です。

```
freq            (for frequency)
mean            (for mean of varname)
sd              (for standard deviation)
sum             (for sum)
rawsum          (for sums ignoring optionally specified weight)
count           (for count of nonmissing observations)
n               (same as count)
max             (for maximum)
min             (for minimum)
median          (for median)
p1              (for 1st percentile)
p2              (for 2nd percentile)
:
p50             (for 50th percentile -- same as median)
:
p98             (for 98th percentile)
p99             (for 99th percentile)
iqr             (for interquartile range)
```

カテゴリー別に連続変数の統計量を得られるという点は2－2.（2）の tabstat と同じ機能です。特徴として、tabstat は同時に出力可能な変数が多いという利点があり、table は表側が表示されるため視覚的に判別しやすい表を出力できる利点があります。

さらに、table では by( ) オプションを指定することで2段階のカテゴリー分類をすることが可能です。

```
. table time, c(mean y sd y mean x1 sd x1) by(yesno)

----------------------------------------------------------
yesno and |
time      |    mean(y)       sd(y)    mean(x1)      sd(x1)
----------+-----------------------------------------------
no        |
        1 |   541.6194    292.3441    239.8083    138.2875
        2 |   534.5977    267.1221     257.675    168.9387
        3 |   527.3891    249.2619    245.2759    108.4163
----------+-----------------------------------------------
yes       |
        1 |   827.9515    480.8857    222.8227    136.9881
        2 |   816.2902    509.9375     231.453    135.9933
        3 |    819.163     481.002    245.4531    183.3931
----------------------------------------------------------
```

（３） データを記述統計量で構成されるデータセットに変換する

collapse コマンドは、データを記述統計量で構成されるデータセットに置き換えます。(そのため、collapse コマンドで指定しなかったデータは全て消失する点に注意が必要です。)

collapse ( ) var

( )内に以下の統計量を指定します。指定のない場合は平均値で計算されます。

statname	definition
mean	means
sd	standard deviations
sum	sums
rawsum	sums ignoring optionally specified weight
count	number of nonmissing observations
max	maximums
min	minimums
median	medians
p1	1st percentile
p2	2nd percentile
:	3rd -- 49th percentiles
p50	50th percentile (same as median)
:	51st -- 97th percentiles
p98	98th percentile
p99	99th percentile
iqr	interquartile range

var の部分に変数名を指定します。また、by( ) オプションによりカテゴリー別に記述統計量を作成できます。ここには複数の変数を指定することが可能です。

ためしに、複数年度の企業別財務データから、企業別の平均値を抽出する方法を見てみましょう。今、データセットには、以下のように、企業の ID 番号(id)、年次(year)、従業者数(labor)、賃金(wage)のデータが含まれているとします。

```
. list fid year labor wage

     +--------------------------------+
     |  fid   year   labor     wage   |
     |--------------------------------|
  1. | 6501   1994   80493   7.459692 |
  2. | 6501   1995   78368   7.559527 |
       （省略）
  6. | 6501   1999   66046   8.325031 |
  7. | 6501   2000   58739   8.417542 |
  8. | 6501   2001   54017   8.820519 |
  9. | 6501   2002   48590   9.560157 |
 10. | 6502   1994   74558   6.690483 |
       （省略）
 14. | 6502   1998   66471   8.003415 |
 15. | 6502   1999   63328   7.752084 |
       （省略）
```

データセットの概要は、

```
. des

Contains data
  obs:         2,910
 vars:             6
 size:        75,660 (99.3% of memory free)
-------------------------------------------------------------------------------
              storage  display     value
variable name   type   format      label      variable label
-------------------------------------------------------------------------------
year            int    %8.0g
fid             long   %12.0g
labor           long   %12.0g
slsprofit       float  %9.0g
wage            float  %9.0g
rdsls           float  %9.0g
-------------------------------------------------------------------------------
Sorted by:
```

ここで、企業ごと(fid)、従業者数、賃金の平均値を求めたいとします。

collapse (mean) labor wage,by(fid)

データセットは以下のような形に変更されます。

```
. des

Contains data
  obs:             4
 vars:             3
 size:            72 (99.9% of memory free)
------------------------------------------------------------------------------
              storage  display     value
variable name   type   format      label      variable label
------------------------------------------------------------------------------
fid             int    %8.0g
labor           double %12.0g                 (mean) labor
wage            float  %9.0g                  (mean) wage
------------------------------------------------------------------------------
Sorted by:  fid
```

確かに、標本数が減少しています。ただし、データセットは置き換わりますが、「Stata Result」画面に表が表示されるわけではありません。統計量を確認するには、別途 list コマンドにより画面表示をするか、browse コマンドによりデータ表示をする必要があります。collapse コマンドと browse コマンドを使うことで、簡単にエクセルなどの表計算ソフトにデータを移し変えることが可能となります。

上記の作業結果を、**list** すると以下のようになります。

```
. list  labor wage

     +--------------------------+
     |       labor     wage     |
     |--------------------------|
  1. | 67156.77778   8.307827   |
  2. | 63760.33333   7.962406   |
  3. | 45268.22222   10.37165   |
  4. | 11891.44444   7.572086   |
     +--------------------------+
```

by()オプションを使う際に、複数の変数を指定することも可能です。たとえば、市区町村別のデータセットがあったとして、個々のデータは都道府県コード(prefecture)と市区町村コード(city)で識別されているとします。

Prefecture	city	Production
1	101	1200
1	101	800
1	102	1921
	（省略）	
2	101	1310
2	104	1050
	（省略）	
3	101	2560
3	102	1420

このよう複数の変数で識別さているデータセットの場合、by オプションで複数の変数を指定します。

collapse (sum) production, by(prefecture city)

また、**collapse** コマンドを使うとデータセット自体が入れ替わってしまいます。そこで、同じデータセットで何度も **collapse** コマンドを使って、複数のデータセットを作成する場合は、処理前にデータを保存し、**collapse** で処理した後に、再度データを呼び出す必要があります。このような場合には、**preserve** コマンドと **restore** コマンドが便利です。

preserve は、データセットをメモリー上に保存（ファイルの上書き・新規作成は行わない）し、**restore** は、**preserve** でメモリー上に保存したデータセットを呼び出してくれます。次の例では、**collaps**e の前後に、**preserve** と **restore** を入れて、**collapse** 後に変更になったデータセットを、**restore** により **collaps**e 以前のデータに復元する処理を確認したものです。

```
. preserve

. collapse (mean) labor wage,by(fid)

. des
Contains data
  obs:             4                          ← collapse により標本数が減少
 vars:             3
 size:            80 (99.9% of memory free)
-------------------------------------------------------------------------------
              storage  display     value
variable name   type   format      label      variable label
-------------------------------------------------------------------------------
fid             long   %12.0g
labor           double %12.0g                 (mean) labor
wage            float  %9.0g                  (mean) wage
-------------------------------------------------------------------------------


. restore          ← preserve 以前のデータセットを復元
. des
Contains data
  obs:            36                          ← collapse 処理の前の標本数に戻る
 vars:             4
 size:           648 (99.9% of memory free)
-------------------------------------------------------------------------------
              storage  display     value
variable name   type   format      label      variable label
-------------------------------------------------------------------------------
year            int    %8.0g
fid             long   %12.0g
labor           long   %12.0g
wage            float  %9.0g
-------------------------------------------------------------------------------


. list
     +--------------------------------+
     | year    fid   labor       wage |
     |--------------------------------|
  1. | 1994   6501   80493   7.459692 |
  2. | 1995   6501   78368   7.559527 |
  3. | 1996   6501   75590   8.030771 |
```

３－３．階級別カテゴリー変数の作成（度数分布表の作成）

度数分布表などを作成する場合、連続変数を階級別のカテゴリー変数（階級値）に置き換える必要があります。たとえば、電機メーカーの財務データを使って、従業員階級別の度数分布表の作成方法を考えましょう。最終的に作成したい表は、以下のような従業員数階級別の企業数を表示した表になります。

従業員数 階級	企業数	Percent	累積
1～99	21	6.29	6.29
100～999	191	57.19	63.47
1000～	122	36.53	100.00
Total	334	100.00	

この表を作成するためには、各企業を従業員数階級ごとに振り分けなければなりません。この作業をオーソドックスに進めるとなると、以下のように **replace** コマンドを繰り返し実行することになります。

```
gen newvar=.
replace newvar=x1 if var<=x1
replace newvar=x2 if var>x1&var<=x2

replace newvar=xn if var>x1&var>x2&var>x3&var>…
```

この作業を **replace** コマンドを使って地道に作業するのはかなり面倒です。そこで、以下の、**recode** 関数を用います。

```
gen newvar=recode(var, x1,x2,x3,…,xn)
```

ただし、**x1<x2<x3<…<xn** とします。
このコマンドは上記ののコマンド群と同義になります。上の具体例のように規模別に 99 人以下、100 人以上 999 人、1000 人以上のカテゴリー変数を作成したい場合、

```
gen labor_category=recode(labor, 99, 999,1000)
```

とします。このコマンドは以下の作業と同じ結果になります。

```
gen labor_category=.
replace labor_category =99 if labor<=99
replace labor_category =999 if labor>99&labor<=999
replace labor_category =1000 if labor>999
```

従業員数のように整数値であれば問題ないですが、比率のように実数値の場合は注意が必要です。たとえば、パート従業員比率のカテゴリーを作成する場合は。

gen part_category=recode(ratio_part, 0.25, 0.5,0.75,1)

とします。このコマンドは以下の作業と同じ結果になります。

```
gen part_category=.
replace part_category =0.25 if ratio_part<=0.25
replace part_category =0.5    if ratio_part>0.25&ratio_part<=0.5
replace part_category =0.75 if ratio_part>0.5&ratio_part<=0.75
replace part_category =1 if ratio_part>0.75
```

例として、電気機器メーカー334 社の従業者数の度数分布表を作成しましょう。
まず、データの記述統計量を sum で確認しましょう。

```
. su labor

    Variable |       Obs        Mean    Std. Dev.       Min        Max
-------------+--------------------------------------------------------
       labor |       334    2416.356    7143.216          4      58739
```

次に、この 334 社のデータを、3 階級の階級値に置き換えた変数を作成します。
. gen labor_category=recode(labor,99,999,1000)

これを tabulate で表示すると、以下のような度数分布表が完成します。

```
. tabulate labor_category

labor_categ |
        ory |      Freq.     Percent        Cum.
------------+-----------------------------------
         99 |         21        6.29        6.29
        999 |        191       57.19       63.47
       1000 |        122       36.53      100.00
------------+-----------------------------------
      Total |        334      100.00
```

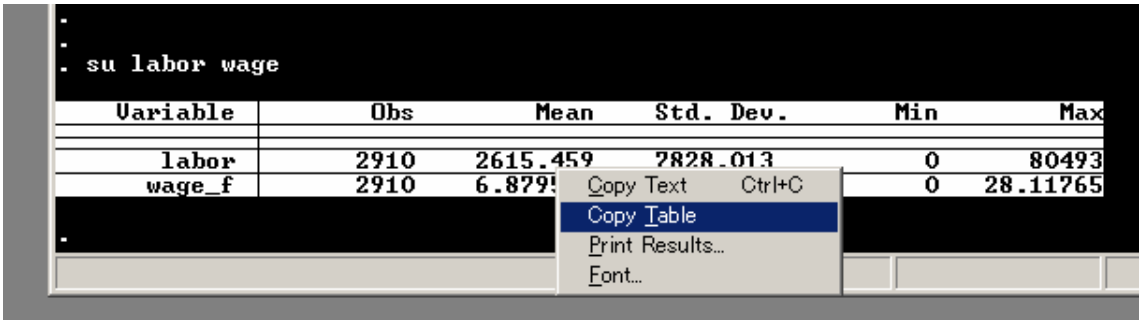
３－４．データのエクセルへの移行

論文を書く際には、Stata で作成した表などを、Result ウインドウのログではなく、EXCEL 等で整形して利用することが多いかと思います。Stata では、結果表や元データの一部を EXCEL に貼り付けたり、、全データシートを EXCEL 形式に変換することができます。

（１）作表結果の貼り付け

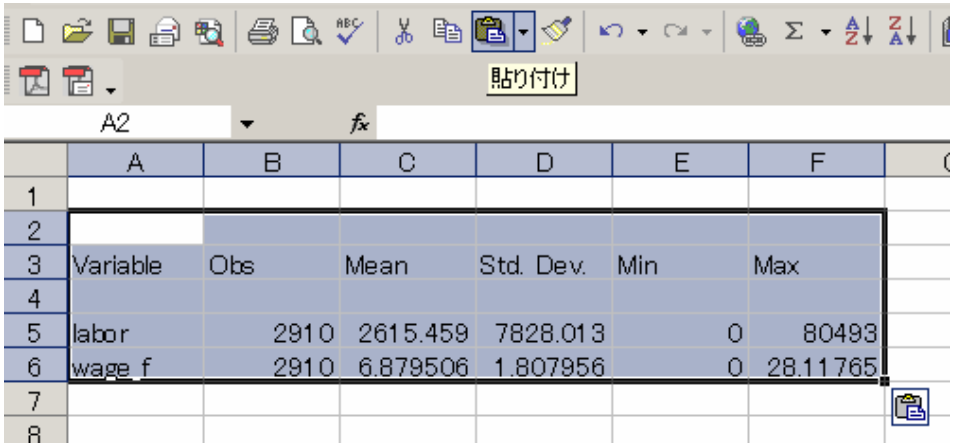
Result Window の画面をカット&ペーストすることで簡単に作表結果を EXCEL に移行することができます。まず、Result Window の結果をマウスで領域指定します。次に、右クリックして、図３－１のように”Copy Table”を選択します。

図３－１



次に、EXCEL を開き、「貼り付け」を行うと、図３－２のように表をそのまま EXCEL 上で復元することができます。

図３－２



（２）データの貼り付け

データの一部を EXCEL に移行させる際、まず **browse** コマンドにより、移行させたいデータを **stata browser** に表示させます。例えば、

browse labor if labor<10000

表示される **stata browser** の範囲を選択しコピーします（図３－３）。EXCEL を開き、「貼り付け」を

行うと EXCEL 上に復元されます（図３－４）。

図３－３

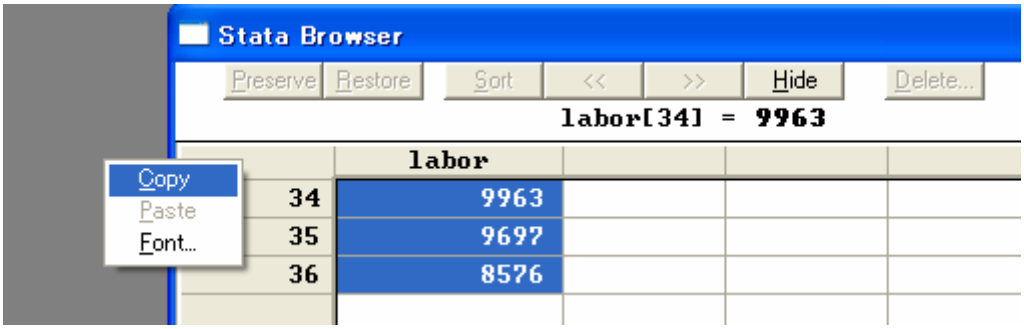
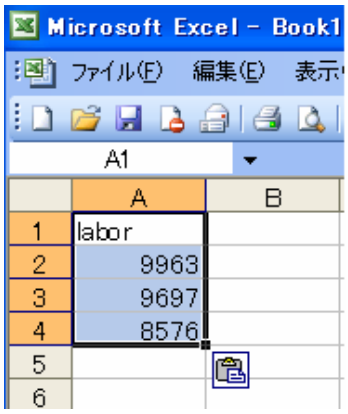


図３－４



（２）データの形式変換

Stata 上でデータを加工した後、その現状の加工済データのまま EXCEL 形式で保存しておきたい時などには **outsheet** コマンドを用いて全データもしくは指定変数系列を形式変換させます。

outsheet using data , replace

↑　　同名の既存ファイルに上書きする場合の指定（指定しない際には “, replace ”を除く）

新たに保存するファイル名

この時、**データはタブ区切り形式で data.out として保存されます。**これを EXCEL 形式まで変換するには、まず EXCEL の「ファイル」の「開く」から**.out** として保存されたファイルを指定します。テキストファイル・ウィザードが開くので「カンマやタブなどの区切り文字によってフィールドごとに区切られたデータ」を指定すると EXCEL 形式でデータを確認できます。

outsheet using data , replace comma

とすると、タブ区切りではなくコンマ区切り形式で、data.out ファイルが保存されます。**replace** 以降に **nonames** を加えると変数コード行を除いたファイルが保存されます。

第４章　回帰分析・離散選択モデルの推定

本節では、回帰分析および離散選択モデルの推定を説明します。ほとんどの回帰分析が

コマンド名 [被説明変数] [説明変数]

の順に並べてリターンキーを押せば結果が出力されます。コマンドによっては、オプションをつけることも可能です。その際は、通常、説明変数の後ろに、カンマをつけてその後ろにオプションを指定します。

コマンド名 [被説明変数] [説明変数], [オプション]

また、サンプルを限定して分析する場合、条件式 if でサンプルを絞ることができます。

コマンド名 [被説明変数] [説明変数] if　condition==1

４－１．回帰分析

本節では、最も単純な最小二乗法（以下、OLS）による回帰分析を説明します。本章の冒頭で説明したとおり、コマンド名 被説明変数 説明変数の順に並べれば回帰分析を行うことができます。最も単純な消費関数を例に挙げて OLS を説明します。
推計式は、

$$Cons_t = \text{定数項} + Y_t + \varepsilon_t$$

です。$Cons_t$ はt期の消費、Y_t は t 期の所得、ε_t 誤差項です。

year	cons	Y
1980	171396.4	312835.2
1981	175753.5	322586.0
1982	183732.8	333273.5
1983	187904.2	341441.8
1984	191204.8	353575.1
1985	199016.7	370527.9
1986	205480.0	379843.8
1987	216162.3	404032.7
1988	226153.3	426670.6
1989	240139.1	451819.8
1990	245054.9	470701.9
1991	251837.0	480778.1
1992	256197.7	482596.4
1993	262698.4	484486.3
1994	270053.6	492857.9
1995	273573.4	505715.3
1996	276604.6	520134.5
1997	276918.2	523999.4
1998	279262.5	516623.9
1999	277907.6	518878.4

ＯＬＳの基本式

reg 被説明変数 説明変数 if 条件式, (option)

この式が、最も基本的な OLS を実行するコマンドです。Stata では、option で指定をしなければ、自動的に回帰式に定数項が含まれます。したがって、何も条件やオプションをつけないで、先の消費関数を推計するコマンドは、

reg cons y

となります。

```
      Source |       SS       df       MS              Number of obs =      20
-------------+------------------------------           F(  1,    18) = 2504.47
       Model |  2.8294e+10     1  2.8294e+10           Prob > F      =  0.0000
    Residual |   203354041    18  11297446.7           R-squared     =  0.9929
-------------+------------------------------           Adj R-squared =  0.9925
       Total |  2.8498e+10    19  1.4999e+09           Root MSE      =  3361.2

------------------------------------------------------------------------------
        cons |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
           y |   .5139885   .0102706    50.04   0.000     .4924108    .5355662
       _cons |   9937.722   4527.128     2.20   0.042     426.5786    19448.86
------------------------------------------------------------------------------
```

Stata では、何も指定しない場合、説明変数に自動的に定数項が含まれてしまいます。定数項を外して推計したい場合には nocons オプションを指定します。

reg cons y, nocons

（１） ラグ付き変数の取り扱い（システム・ファンクションの利用）

先述の推計式にラグ付き変数を含める場合、例えば

$$Cons_t = \text{定数項} + Y_t + Y_{t-1} + \varepsilon_t$$

とする場合、変数 Y の１期ラグ付き変数が必要となります。この時、システム変数[_n-1]を利用するとよいでしょう。変数 Y_{t-1} を、以下のように作成し、上式を推定することができます。

gen y1=y[_n-1]
reg cons y y1

（２） 質的変数の取り扱い

回帰分析においては、質的な情報を扱う際には、その変数をそのまま用いるのではなく、ダミー変数と呼ばれる 0/1 の変数に置き換えて分析されることがしばしばあります。単純なダミー変数であれば、たとえば、性別の違いを分析に取り込みたい場合、以下のような手順を踏みます。データセットでは、性別は、sex(1のとき男性、2 は女性)となっているとすると、

```
gen d_male=0
replace d_male=1 if sex==1
reg wage age education d_male
```

となります。d_male は男性のとき1を示す変数です。この係数は、賃金の男女差を示すことになります。

なお、ダミー変数を作成する2つのコマンドは、以下の一文にまとめることもできます。

```
gen male=sex==1
```

連続変数からダミー変数を作成する場合は、まず、38 ページで説明した方法でカテゴリー変数を作成します。次に、新たに作成したカテゴリー変数（ここでは、labor_category としましょう。）をもとにダミー変数を作成するには、以下のようなコマンドを使います。

```
tabulate labor_category, generate(empcat)
```

このコマンドにより、empcat1, empcat2, empcat3, empcat4 の4つの変数が生成されます。
37 ページの例と同じデータセットでダミー変数を作成してみましょう。

```
. tab labor_category,generate(empcat)

labor_categ |
        ory |      Freq.     Percent        Cum.
------------+-----------------------------------
         99 |         21        6.29        6.29
        999 |        191       57.19       63.47
       1000 |        122       36.53      100.00
------------+-----------------------------------
      Total |        334      100.00
```

describe で確認すると、新しい変数が生成されていることがわかります。

```
. des

Contains data
  obs:           334
 vars:             6
 size:        11,022 (99.9% of memory free)
-------------------------------------------------------------------------------
              storage  display     value
variable name   type   format      label      variable label
-------------------------------------------------------------------------------
id              long   %12.0g
labor           long   %12.0g
labor_category  float  %9.0g
empcat1         byte   %8.0g                  labor_category==    99.0000
empcat2         byte   %8.0g                  labor_category==   999.0000
empcat3         byte   %8.0g                  labor_category==  1000.0000
-------------------------------------------------------------------------------
```

新しい変数 empcat1,empcat2, empcat3 は 0、1で構成されていることがわかります。

```
. sum empcat*

    Variable |       Obs        Mean    Std. Dev.       Min        Max
-------------+--------------------------------------------------------
     empcat1 |       334    .0628743    .2431008          0          1
     empcat2 |       334    .5718563    .4955521          0          1
     empcat3 |       334    .3652695    .4822281          0          1
```

ここで、作成した企業規模ダミー変数を回帰分析で取り扱うには、

reg y x1 x2 x3 empcat1 empcat2

とします。ダミー変数は、すべて説明変数に挿入すると、定数項と多重共線性を引き起こしうまく推定できないので、ここでは、empcat3 を省いています。

Stata では、質的変数をダミー変数に自動的にダミー変数に置き換えて回帰分析を実行するコマンドも備え付けてあります。ただし、この方法は、標準的な最小二乗法(reg コマンドによる分析)にしか用いることが出来ませんので、注意が必要です。

（３）xi:reg コマンド

基本式は以下のようになります。

xi:reg 被説明変数 説明変数 i.カテゴリー変数

コマンドとして、xi:reg を入力し、ダミー変数を作成したいカテゴリー変数の前に i. をつけます（i の後にピリオドを忘れないよう注意）。
このコマンドは、質を表すカテゴリー変数に対して、自動的にカテゴリー毎のダミー変数を作成してくれるコマンドです。具体例として、以下の推計式を考えます。

賃金(wage)=定数項+年齢(age) +大学院卒ダミー(D[education=1])
+大学卒ダミー(D[education=2])
+短大卒ダミー(D[education=3])
+高卒ダミー(D[education=4])

ここで、education は、大学院卒なら1、大学卒なら2、短大卒なら3、高卒なら4を示すカテゴリー変数であるとします。このカテゴリー毎にダミー変数を作って説明変数に加えたい場合、xi:reg を用いると自動的にカテゴリー毎にダミー変数を作って推計してくれます。今回のケースであれば、以下のようにコマンドを入力します。

xi:reg wage age i.Education

推計結果は以下のように表示されます。

```
i.education       _Ieducation_1-4     (naturally coded; _Ieducation_1 omitted)

      Source |       SS       df       MS              Number of obs =      54
-------------+------------------------------           F(  4,     49) =   24.39
       Model |  8.92978039     4  2.2324451            Prob > F      =  0.0000
    Residual |  4.48509126    49  .091532475           R-squared     =  0.6657
-------------+------------------------------           Adj R-squared =  0.6384
       Total |  13.4148716    53  .253110786           Root MSE      =  .30254

------------------------------------------------------------------------------
        wage |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
         age |   .0263066   .003927     6.70   0.000     .0184149    .0341982
_Ieducatio~2 |  -.5410875   .144381    -3.75   0.000     -.831232    -.250943
_Ieducatio~3 |  -.7688748   .1875269   -4.10   0.000    -1.145724   -.3920254
_Ieducatio~4 |    -.98413   .1953334   -5.04   0.000    -1.376667   -.5915928
       _cons |   6.009725   .2079137   28.90   0.000     5.591907    6.427544
------------------------------------------------------------------------------
```

この計算結果は、Edum1 は大学院卒＝1、その他＝0 のダミー変数（Edum2 は、大学卒=1、その他=0 のダミー、以下続く）としたおときに、

reg wage age Edum1 Edum2 Edum3 Edum4

という回帰式と同じ結果をもたらします。

（４）areg コマンド

xi:reg の類似のコマンドとして、areg コマンドがあります。xi:reg コマンドを用いると、すべてのダミー変数の係数が表示されますが、必ずしもダミー変数の係数が必要でない場合があります。その際、areg コマンドを用いると、同じ計算を Speedy に実行してくれます。賃金を従業員の年齢、学歴で分析する例を見ましょう。先の例を用いて推計する場合には、以下のようにコマンドを入力します。

areg 被説明変数 説明変数, absorb(カテゴリー変数名)

xi:reg コマンドとの違いは、自動的に作成されたダミー変数の個々の係数パラメータの値や t 値などを推計結果として表示しない点です。
推計結果は以下のように表示されます。

```
                                                Number of obs =      54
                                                F(  1,    49) =   44.87
                                                Prob > F      =  0.0000
                                                R-squared     =  0.6657
                                                Adj R-squared =  0.6384
                                                Root MSE      =  .30254

------------------------------------------------------------------------------
       lwage |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
         age |   .0263066    .003927     6.70   0.000     .0184149    .0341982
       _cons |   5.452406   .1475945    36.94   0.000     5.155804    5.749009
-------------+----------------------------------------------------------------
   education |          F(3, 49) =      9.393   0.000           (4 categories)
------------------------------------------------------------------------------
```

推計結果の一番下に出ている F 検定の結果は、Edum1 ～ Edum4 のパラメータが同時に 0 になるかどうかを検定した結果を表しています。**xi:reg** コマンドでは、F 検定が行われない代わりに、個別のダミー変数のパラメータや t 値が表示されています。説明変数として用いた age の係数パラメータが **areg** コマンドを用いた場合と同じになることを確認してください。

４－２．離散選択モデル

Stata は個票データ処理に強みを発揮しますが、個票データの中には、アンケート調査のようなデータが使われている場合も見受けられます。本節では、そのようなデータを分析する離散選択モデル（質的変量モデル）を紹介します。

（１）プロビット分析

質的データを被説明変数とするモデルの代表的な分析手法がプロビット分析です。プロビットモデル、ロジットモデルは回帰分析の考え方を応用した確率モデルに基づく分析手法であるため、本章の冒頭で説明したとおり、コマンド名 被説明変数 説明変数の順に並べれば分析を行うことができます。プロビット分析では、以下のコマンドを用いて分析します。

probit 被説明変数 説明変数

以下では、50 人の既婚女性の労働に関するデータを例にして説明します。
推計式は以下のようなものを想定します。

Work = C18 + AGE + AGE^2 + ED + HI

変数の説明は以下のとおり。
Work ： 0＝就労していない、1＝就労している。
C18 ： 18 歳未満の子供の数
AGE ： 年齢
ED ： 教育年数
HI ： 夫の収入

このモデルをプロビット分析する場合、以下のようにコマンドを入力します。

probit Work C18 Age Age2 ED HI

推計結果は以下のように表示されます。

```
Iteration 0:   log likelihood =  -32.67091
Iteration 1:   log likelihood = -26.474454
Iteration 2:   log likelihood = -26.229633
Iteration 3:   log likelihood = -26.227426
Iteration 4:   log likelihood = -26.227426

Probit estimates                                  Number of obs   =         50
                                                  LR chi2(5)      =      12.89
                                                  Prob > chi2     =     0.0245
Log likelihood = -26.227426                       Pseudo R2       =     0.1972

------------------------------------------------------------------------------
        work |      Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
         c18 |  -.4013713   .1897784    -2.11   0.034    -.7733302   -.0294125
         age |   .1501527   .1375394     1.09   0.275    -.1194196    .419725
        age2 |  -.0023274   .0015347    -1.52   0.129    -.0053352    .0006805
          ed |   .0536939   .0906034     0.59   0.553    -.1238854    .2312732
          hi |  -4.96e-06   .0000102    -0.48   0.628     -.000025    .0000151
       _cons |  -1.478264    2.82501    -0.52   0.601    -7.015182    4.058654
------------------------------------------------------------------------------
```

通常の回帰分析では、係数は、説明変数が1増えると被説明変数がどの程度変化するか、という限界効果として解釈できますが、probit モデルの場合、そのような解釈はできません。probit モ

デルで限界効果を導くには、通常、多少の計算を必要としますが、Stata では、dprobit コマンドをつかってプロビットモデルにおける限界効果を表示することができます。コマンドはプロビット分析と同じく以下のように書きます。

dprobit Work C18 Age Age2 ED HI

さらに、Stata では順序プロビットモデルも oprobit コマンドを使って推計することができます。順序プロビットモデルの場合も同様に

oprobit Work C18 Age Age2 ED HI

と書くことになります。

（２）ロジット分析

プロビット分析では確率分布として正規分布を用いてきましたが、ロジスティック分布を用いるロジット分析も質的変量データの分析にしばしば用いられます。先ほどの例に対してロジット分析を行う場合、以下のようなコマンドを入力します。

logit Work C18 Age Age2 ED HI

推計結果は以下のように表示されます。

```
Iteration 0:   log likelihood =  -32.67091
Iteration 1:   log likelihood =  -26.42873
Iteration 2:   log likelihood = -26.250084
Iteration 3:   log likelihood = -26.248375
Iteration 4:   log likelihood = -26.248374

Logit estimates                                   Number of obs   =         50
                                                  LR chi2(5)      =      12.85
                                                  Prob > chi2     =     0.0249
Log likelihood = -26.248374                       Pseudo R2       =     0.1966

------------------------------------------------------------------------------
        work |      Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
         c18 |  -.6613577   .3264931    -2.03   0.043    -1.301272    -.021443
         age |   .2636399   .2363032     1.12   0.265    -.1995059    .7267857
        age2 |  -.0040124    .002665    -1.51   0.132    -.0092357    .0012109
          ed |   .0812957   .1520127     0.53   0.593    -.2166436    .3792351
          hi |  -8.43e-06   .0000175    -0.48   0.629    -.0000427    .0000258
       _cons |  -2.659551   4.717939    -0.56   0.573    -11.90654    6.587439
------------------------------------------------------------------------------
```

４－３．回帰分析結果の整理（outreg コマンド）

複数の回帰分析結果を journal スタイルでまとめるのは、結構面倒な作業です。こんなとき、**outreg** コマンドを用いると便利です。

outreg コマンドは、ado ファイルで提供されています。まず、以下の WEB ページから、outreg.ado ファイルをダウンロードしてください。
http://ideas.repec.org/c/boc/bocode/s375201.html

※ internet explorer にプログラムが表示されたら、そのページを「テキスト形式」で「名前をつけて保存」してください。さらに、拡張子を".ado"に変更してください。
ダウンロードしたファイルは、stata をインストールしたときに生成される"ado"フォルダーの下の"personal"フォルダーに移してください。
"ado"ファイルの使い方は、User's Guide も参考にしてください。

reg コマンドの実行後に、outreg using filename.doc と分析結果の出力先を指定すると、filename.doc というファイルが生成さます。
次の例では、県民経済計算（経済企画庁・平成２年）の 47 都道府県の貯蓄額(save)と所得(income)を使った回帰分析の結果を outreg コマンドによって、save.doc ファイルに出力しています。

*回帰式①

```
. reg save income

      Source |       SS       df       MS              Number of obs =      47
-------------+------------------------------           F(  1,    45) =   71.47
       Model |  109948399      1  109948399            Prob > F      =  0.0000
    Residual |  69224547.9     45  1538323.29          R-squared     =  0.6136
-------------+------------------------------           Adj R-squared =  0.6051
       Total |  179172947     46  3895064.06           Root MSE      =  1240.3

------------------------------------------------------------------------------
        save |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
      income |   3.756302   .4443138     8.45   0.000     2.861408    4.651196
       _cons |  -3866.052   1116.157    -3.46   0.001    -6114.107   -1617.997
------------------------------------------------------------------------------

. outreg using save.doc
```

*回帰式②

```
. reg save

      Source |       SS       df       MS              Number of obs =      47
-------------+------------------------------           F(  0,    46) =    0.00
       Model |           0     0           .           Prob > F      =       .
    Residual |   179172947    46  3895064.06           R-squared     =  0.0000
-------------+------------------------------           Adj R-squared =  0.0000
       Total |   179172947    46  3895064.06           Root MSE      =  1973.6

------------------------------------------------------------------------------
        save |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
       _cons |    5445.34   287.8779    18.92   0.000     4865.872    6024.809
------------------------------------------------------------------------------

. outreg using save.doc,append
```

こうして生成された save.doc ファイルの中身は以下のとおりです。

```
        (1)      (2)
        save     save
income  3.756
        (8.45)**
Constant        -3,866.052       5,445.340
        (3.46)**         (18.92)**
Observations    47       47
R-squared       0.61     0.00
Absolute value of t statistics in parentheses
* significant at 5%; ** significant at 1%
```

これをコピーしてEXCELに貼り付けると、以下のような journal フォーマットの表が得られます。

	(1)	(2)
	save	save
income	3.756	
	(8.45)**	
Constant	-3,866.05	5,445.34
	(3.46)**	(18.92)**
Observations	47	47
R-squared	0.61	0

Absolute value of t statistics in parentheses
* significant at 5%; ** significant at 1%

第５章　パネルデータによる分析

５－１．パネルデータとは

パネルデータとは、同一の主体/個体（個人、家計、企業など）を複数の時点について観測したものです。STATA では、個体を認識する変数を行方向に並べ、個体ごとの同一変数の異時点の観測値が列方向・行方向のどちらに並ぶかにより、データの構成が大きく異なります。

LONG 形式:複数の個体のデータの集合が縦方向に接続されたデータ
WIDE 形式:複数の個体のデータ系列が横方向に接続されたデータ

LONG 形式のデータ（例）

fid	year	labor	slsprofit	head-q
6501	1994	80493	.0173707	13
6501	1995	78368	.0214952	13
6501	1996	75590	.031215	13
6501	1997	72193	.0195598	13
6501	1998	70375	.0042226	13
6501	1999	66046	-.0303931	13
6501	2000	58739	.0084272	13
6501	2001	54017	.0139593	13
6501	2002	48590	-.0231846	13
6502	1994	74558	.016504	14
6502	1995	73463	.021515	14
6502	1996	71170	.0326982	14
6502	1997	68441	.0253295	14
（以下省略）				

WIDE 形式のデータ（例）

fid	labor1994	labor1995	labor1996	labor1997	labor...
6501	80493	8368	5590	2193	…
6502	74558	3463	1170	8441	…
6503	49842	8421	7752	7372	…
6504	14094	3794	3202	2870	…

Long 形式のデータ作成

同一個体を追跡調査している調査統計に対し、同一変数に関する一連の調査結果であっても、時点ごとに個別データシートが存在する場合があります。このような時、append コマンドによりパネルデータセットを構築することができます。ただし、各データシートにおいて、時点を識別する変数が各シートに含まれている必要があるのに注意が必要です。append コマンドの扱いは『２－１．データの縦方向の結合』をご参照ください。

Wide 形式のデータ作成

Wide 形式では、各個体の識別変数に対応して、全変数が横に並ぶことになります。そのため、新たなデータセットの追加などには、merge コマンドにより対応することができます。merge コマンドの扱いは『２－２. データの横方向の結合』を参照してください。

Long 形式と Wide 形式の特性

パネル計量分析を行うには、データセットが必ず Long 形式となっている必要があります。ただし、データの扱いは、Wide 形式である方が便利な時もあります。例えば、GDP 主要項目の GDP 成長率への寄与度を算出したい場合、異なる変数の異なる時点を抽出して計算する必要があります。

id	year	GDP	C	I	･･･
0001	:	(省略)	:	:	:
0001	1999	a	b	c	:
0001	2000	d	e	f	:
0002	:	(省略)	:	:	:
0002	1995	g	h	i	:
0002	2000	j	k	l	:
（以下省略）					

ここで、2000 年の C(民間最終消費支出)の GDP 成長率への寄与度を測るとき、4-1.(1)のシステムファンクション[_n-1]を用いて、

```
gen new_var_name == ( C － C[_n-1])/GDP[_n-1] if year==2000
```

と計算式を指定することができます。この時、固体 0001 については、

2000 年消費の GDP 成長率寄与度＝ (e － b)÷ a

として正しく計算されることになります。しかし個体 0002 は 1996 年から 1999 年のデータが欠損していることから、(k-h)/g が計算されます。これは対 1995 年変化率を算出していることになり、本来示されるべき値(欠損値".")を得ることができません。後述の５－１.(３)で紹介するデータ・オペレータ・ファンクションにより個体別に正しく計算することが可能ではありますが、複雑な計算式の場合や、全期間に対して時系列の寄与度データが必要なのではなく、ある一時点の寄与度のみ抽出した場合などは、データセットが Long 形式ではなく、Wide 形式となっていると便利です(データ形式の変換は、５－１.(１)をご参照ください)。

id	･･･	GDP1999	GDP2000	･･･	C1999	C2000	･･･
0001	:	a	b	･･･	c	d	･･･
0002	:	e	f	･･･	g	h	･･･
（以下省略）							

上のようにデータセットが Wide 形式の時、

```
gen new_var_name = ( C2000 － C1999)/GDP1999
```

で、個体別に 2000 年の正しい「消費の GDP 成長率寄与度」を計れることとなります。

balance パネルと unbalance パネル

balanced panel とは、使用するデータセットの各個体の変数が全期間揃っている(欠損値を含

まない)パネルデータセットであることを言います。反対に、ある個体のある時点のデータが欠損している場合は unbalanced panel と言います。

（１）パネルデータ形式を変換する

パネルデータの『 LONG 形式 ⇔ WIDE 形式 』変換を reshape コマンドにより行うことができます。「５－１. パネルデータとは」のデータ形式別のパネルデータセット例を用いて見てみましょう。

【long 形式 ⇒ wide 形式】の変換

reshape wide labor slsprofit, i(fid) j(year)
変換対象の変数

wide に続いて変換したい変数名を記入します。個体ごとに時間を通じて一定の変数(たとえば、表の変数のうち、”head-q” のように個体ごとにみると、一定になっている変数)は記入する必要はありません。ただし、個体により、時間により異なる値をもつ変数がデータセットに含まれている(表の labor や slsprofit のような変数)にも関わらず、変換対象の変数として記述から漏れている時、データ形式変換は行われずエラー表示が返されます。コマンドライン中の wide 以下には time variant(時間について可変)な変数は全て記入するようにしましょう。

変換の軸となる個体を表わす変数 fid と、時間を表わす変数 year の全データが、一対一の関係であれば問題なく変換されます。誤植などにより重複してデータが存在する場合(例えば fid 番号 5948 の 1999 年のデータが1つ以上存在する場合など)は変換されず、

```
year not unique within fid;
there are multiple observations at the same year within fid.
Type "reshape error" for a listing of the problem observations.
r(9);
```

のようなエラーが表示されます。このような場合の対処方法は、第5章の補論を参照してください。

なお、unbalanced panel である場合、データ変換に特に問題は生じません。欠損しているデータについては「.」の欠損を表わす記述が自動的に置き換わります。

【wide 形式⇒long 形式】の変換

reshape long labor slsprofit, i(fid) j(year)

変換するデータセットには、変換対称として指定する変数名(ここでは labor, slsprofit)と、その変数名に数値が続く変数(ここでは labor1994, labor11995,…)が存在する必要があります。指定変数名に続く数値が、 **j( )** で指定した時間軸変数の値として変換されます。全ての変数が正しく存在する時(共通した変数名があり、その各変数名に共通した数値系列が続いている場合)、細かい指定を省略し **reshape long** と記入するだけで、データ形式が変換されます。

（２）パネルデータとしての認証

パネルデータによる分析を行う際、STATA にデータセットがパネルデータであるという情報を伝える必要があります。

tsset　var1　var2
「var1」には主体を表わす変数名を、「var2」には時間軸を表わす変数名を記述します。

```
. tsset fid year
       panel variable:  fid, 1909 to 359059
        time variable:  year, 1994 to 2002
```

パネルデータであることを伝えたら、パネルデータの形状を xtdes コマンドにより確認できます。

```
. iis fid
. tis year
. xtdes                                                          ①
                                                                 ↓
     fid:  1909, 1993, ..., 359059                       n =       334
    year:  1994, 1995, ..., 2002                        T =         9
           Delta(year) = 1; (2002-1994)+1 = 9
           (fid*year does not uniquely identify observations)

Distribution of T_i:   min      5%     25%       50%       75%     95%     max
                         1       7       9         9         9       9      27
                                                   ↑
     Freq.   Percent    Cum.  |  Pattern            ②
 ----------------------------+-----------      ←③
      256     76.35   76.35  |  111111111    ←94-02 まで連続している標本が 256 社
       27      8.08   84.43  |  .11111111
       16      4.79   89.22  |  11111111.
       12      3.59   92.81  |  ..1111111
       10      2.99   95.81  |  ...111111
        3      0.90   96.71  |  1111111..
        1      0.30   97.60  |  ........1
        1      0.30   97.90  |  ....11111
        7      2.10  100.00  |  (other patterns)
 ----------------------------+-----------
      334    100.00          |  XXXXXXXXX
```

① ここには、個体識別変数(fid)が 1909～359059 までの値の 334 社のデータが、1994～2002 年の 9 時点分あることを示しています。また、変数 fid と year が一対一の関係でないことも(fid*year does not uniquely identify observations)で示しています。そのため、データの重複を修正しなくては、Wide 形式に変換することも回帰分析することもできないことが分

かります。

② ここには、データの欠損に関する情報が得られます。95%のデータは9時点のデータがあることを示していますが、5%のデータは 7 時点のデータであることが示されています。よって、このデータセットは unbalanced panel であることが分かります。

③ ②の情報を、より詳しく示しています。256 サンプルはデータは全期間連続しており、27 サンプルは1期目のデータが欠損していることを示しています。Pattern の列にしめされる「1」はデータ存在していることを示し、「.」はデータが存在していないことを示しています。

（3）データ・オペレータ・ファンクション

tsset の設定により STATA が時系列の概念を認識できるようになると、遅延演算子などのオペレーション・ファンクションを利用することが可能となります。

l. ファンクション 時系列方向のデータを含むデータを扱う際、l. を変数の前に付けることでラグ付変数として認識されます。

```
labor     ≡labor(t)
l.labor   ≡labor(t-1)
l2.labor  ≡labor(t-2)
  :          :
```

f. ファンクション f. を変数の前に付けることで一期前の値を参照します。

```
f.labor   ≡labor(t+1)
f2.labor  ≡labor(t+2)
  :          :
```

d. ファンクション d. を変数の前に付けると、前期値との差分変数として認識します。

```
d.labor   ≡labor(t)-labor(t-1)
```

これらのオペレーション・ファンクションは、個体ごとの時系列を参照して算出されます。その点が変数システムファンクション[_n-1]などと異なり、パネルデータを扱う際の極めて利便性の高いファンクションと言えます。以下に、l. ファンクションとシステムファンクション[_n-1]との違いを例示しましょう。

```
. tsset fid year
. gen test1=l.labor
. gen test2=labor[_n-1]
. list
     +-------------------------------------+
     |   fid   year   labor   test1   test2 |
     |-------------------------------------|
  1. |  1909   1995     535       .       . |
  2. |  1909   1996     529     535     535 |
     |    （省略）                         |
  6. |  1909   2000     470     509     509 |
  7. |  1993   1994     922       .     470 |   ←test2 では個体変数別に
  8. |  1993   1995     929     922     922 |     データが作成がされない
     |    （省略）                         |     様子がわかります。
 15. |  1993   2002     773     799     799 |
 16. |  4062   1994    1450       .     773 |
 17. |  4062   1995    1921    1450    1450 |
          （以下省略）
```

データ・オペレータ・ファンクションにより Long 形式でもデータの扱いが容易になりますが、５－２. で紹介する回帰分析に、オペレータ・ファンクション付の変数を直接組み込むことはできません。回帰分析でラグ付変数などを使用したい場合は、まず一度 gen コマンドで新たな変数を作成し、その新変数を使って回帰分析を試みましょう。

５－２．パネルデータによる回帰分析

パネル計量分析を行う際、データの特性(i:個体を表わす変数、t:時間を表わす変数)に関する情報が必要です。５－１.(2)で指定した tsset から変更がなければ、回帰分析を行うコマンドラインごとに i や t を指定する必要はありません。ただし、データを加工したことで、新たな個体認識変数や時間変数が作成された場合などは、データ特性が変更された情報を STATA に伝えなければなりません。

```
iis varname
tis varname
```

iis コマンドは新たな個体認識の変数の指定、tis は新たな時間変数の指定を行います。この時、tsset で伝えていた情報は残されないため、元の特性を用いて分析し直したい時には、特性変数の再指定をする必要があります。

以下では、実際に回帰分析を行う手順を概説します。ここでは、説明変数に強外生性を仮定し、OLSにより一致推定量を得られるものとして固定効果モデルと変量効果モデルを紹介します。説明変数に内生変数が含まれる場合などや、ダイナミックなモデルを想定する際に操作変数法(xtivreg)などによる推定を行うことがありますが、詳しくは各自マニュアルをご参照ください。なお、パネル分析におけるGMM推定量や、より高度な推定量などは、STATA にプログラムが内蔵されていなくても、研究者などが個人的に作成したプログラムを一般公開している場合もありますの

で、すぐに諦めずに一度
< http://www.stata.com/links/resources2.html >で検索してみることをお勧めします。

（１）線形回帰分析（変量効果モデル、固定効果モデルなど）

```
xtreg depvar indepvar ,xx
```

「depvar」部分に被説明変数を、「indepvar」部分に被説明変数（複数記入可）を記入します。「,xx」の xx 部分には、以下の得たい推定量を記入します。無記入の場合は変量効果モデル re が推定されます。

```
be      between-effects estimator
fe      fixed-effects estimator
re      GLS random-effects estimator
```

（２）ハウスマン検定

STATA には、固定効果モデルと変量効果モデルの推定量を比較して、個体効果が説明変数と相関をもつかどうかのハウスマン検定を以下の手順で行うことができます。

```
xtreg depvar indepvar, fe
est store fixed
xtreg depvar indepvar, re
hausman fixed .
```

なお、下線部分には適当な変数名を指定します。

（３）非線形回帰分析

ここでは、非線形回帰モデルとして、プロビット、ロジットとトービット・モデルのコマンドだけ簡単にご紹介します。オプションなどの詳しい解説はここでは省略しますので、必要に応じてマニュアルをご参照ください。

プロビット・モデル

xtprobit depvar indepvar, i(id)

ロジット・モデル

xtlogit depvar indepvar, i(id)

トービット・モデル

xttobit depvar indepvar , i(id) ll(#)

ここで ll(#)は左に切断されたデータを意味し、# に切断点を記入します。右に切断されたデータの場合は ul(#)を記入します。右にも左にも切断されていたデータを推定するには、ll(#)と ul(#)を両方記入しましょう。

第５章 補論 重複データの対処法

下記のデータのように、1991 年の id=5 のデータのように一つのデータセットの中に 2 つのデータが入っている場合を考えて見ましょう。

id	year	value
1	1990	100
2	1990	100
3	1990	100
4	1990	100
5	1990	100
1	1991	110
2	1991	111
3	1991	112
4	1991	113
5	1991	114
5	1991	114

このデータを無理やり、パネルデータとして認識させようとしても、

```
. tsset id year
repeated time values within panel
```

というメッセージが返ってきます。また、reshape で wide データに変換しようとすると、

```
. reshape wide value,i(id) j(year)
(note: j = 1990 1991)
year not unique within id;
there are multiple observations at the same year within id.
Type "reshape error" for a listing of the problem observations.
r(9);
```

というエラーメッセージが返ってきます。ここで、reshape error と入力すると、id=5 が重複していることがわかります。

```
. reshape error
(note: j = 1990 1991)

i (id) indicates the top-level grouping such as subject id.
j (year) indicates the subgrouping such as time.
The data are in the long form;  j should be unique within i.

There are multiple observations on the same year within id.
```

```
The following 2 out of 11 observations have repeated year values:

      +-----------+
      | id   year |
      |-----------|
  10. |  5   1991 |
  11. |  5   1991 |
      +-----------+

(data now sorted by id year)
```

こういった問題への対処法としては、EXCEL 等で作成した元のデータセットに戻って作成方法に間違いがなかったかを調べるか、duplicatees コマンドを用いて重複しているデータの片方を強制的に削除してしまう方法が考えられます。
duplicates コマンドは、report オプションをつけると、重複状況を表示させることができます。”copies”1となっている行は重複のないデータの数、2は重複するペアの数が表示されます。

```
. duplicates report id year

Duplicates in terms of id year

--------------------------------------
   copies | observations       surplus
----------+---------------------------
        1 |            9             0
        2 |            2             1
--------------------------------------
```

重複しているペアの片方を削除するには、drop オプションを使います。

```
. duplicates drop id year,force

duplicates in terms of id year

(1 observation deleted)
```

このコマンドの後に、データセットを browse すると、次の表のように重複データが強制的に削除差入れていることがわかります。

id	year	value
1	1990	100
1	1991	110
2	1990	100
2	1991	111
3	1990	100
3	1991	112
4	1990	100
4	1991	113
5	1990	100
5	1991	114

第６章　サバイバル分析

６－１．サバイバル分析とは

サバイバル分析とは、誤解を恐れずに言えば、分析上、興味のあるイベントの発生の有無を表す変数と、そのイベントが発生するまでの時間を表す変数との関係を分析する手法です。この手法は、生物学の分野で応用・開発が進められたものですが、近年では事業所の存続・閉鎖に関する分析など、経済学へも応用されています。同様の研究テーマに用いられるその他の手法としては、存続・退出の2者択一によるプロビット・モデルが挙げられますが、プロビット・モデルを用いた分析の場合、いつ参入したかといった過去の履歴が考慮できないという欠点があり、その欠点を補う目的でサバイバル分析が用いられます。

６－２．サバイバルデータとしての認証

STATA においてサバイバル分析を行う場合には、パネル分析と同様、まずサバイバル分析を行うことを STATA に認識させる必要があります。サバイバル分析に用いられるデータには、大きく分けて以下の二つがあります。

Survival-time data:観察された個体の ID、期間を表す変数、failure or censoring を示す変数の三つの要素が入ったデータ。
Count-time data　:Survival-time data の集計版。failure or censoring を示す変数、時点 t における failure or censoring であった個体総数の二つの要素が入ったデータ。

Survival-time data を用いる場合は stset コマンド、Count-time data を用いる場合は ctset コマンドを用いて STATA に認識させます。

stset [timevar], fail(failvar)
ctset [timevar], fail(failvar)

[timevar]には時間を表す変数を、(failvar)には分析上興味があるイベントを表すダミー変数(failure=1,cencsoring=0)を、それぞれ指定します。サバイバル分析では、(failvar)で指定した変数を非説明変数として認識します。

stset を使用した場合には、様々なオプションが利用可能です。例えば、origin(time originvar)と指定すると、イベントが発生するまでの時間(t)をt＝timevar-originvar として計算してくれます。

origin の他にも様々なオプションがありますが、ここでは説明を省略します。各自 STATA マニュアル(Version.8 なら、“Survival analysis and epidemiological tables”)を参照してください。

ここで、企業倒産をイベント(failvar)とする以下のような Survival-time data を考えてみましょう。

id	year	died	closeyear	origin	slsprofit	llabor	wage_f
1	1996	0	2002	1925	0.0131807	9.488124	7.327905
1	2000	0	2002	1925	0.0029958	9.206634	7.818227
2	1996	0	2002	1961	0.0131807	9.488124	7.327905
2	2000	0	2002	1961	0.0029958	9.206634	7.818227
(省略)							
110	1996	0	2001	1970	0.0428923	8.050385	7.597767
110	2000	1	2001	1970	0.0047661	7.959975	8.029329
(省略)							
339	1996	0	2001	1969	0.0264739	7.187657	6.518518
339	2000	1	2001	1969	0.0156754	7.25347	6.512385
(以下省略)							

「id」は企業 id、「year」はデータ年次、「died」は企業倒産の有無を表すダミー変数(倒産＝1)、「closeyear」は倒産年次、「origin」は設立年次、「slsprofit」は売上高利益率、「llabor」は従業員数の対数値、「wage_f」は平均賃金の対数値です。

この Survival-time data を STATA に認識させるため、以下のように指定します。

stset closeyear, fail(died) origin(time origin)

オプションで origin(time origin)と指定しているので、企業倒産と企業の生存年数(closeyear-origin)の関係を分析することを STATA に認識させたことになります。

６－３．サバイバル分析

分析上、興味のあるイベントが少なくとも t 期間以降に発生する確率を示す関数を、生存関数(あるいはハザード関数)と呼びます。サバイバル分析では、ハザード率(次の瞬間に分析上興味があるイベントが起こる確率)を被説明変数、ハザード率に影響を与える変数を説明変数とし、この生存関数(あるいはハザード関数)がどのような要因によって変化するかを推定するものです。STATA では複数の推定方法に対してコマンドが用意されています。

(1)Cox の比例ハザードモデル(Cox Proportional Hazard Models)

詳しい説明は他の教科書に委ねますが、ハザード関数にはベースライン・ハザードと呼ばれる様々な要因を取り除いた場合の全サンプルに共通するハザード率が含まれます。Cox は、ベースライン・ハザードの分布が推定に影響しないような推定方法を考案しました。そのため、ベースライン・ハザードの分布の形を特定せずにハザード関数を推定するモデルを Cox の比例ハザードモデル(Cox Proportional Hazard Models)と呼びます。

Cox の比例ハザードモデルを推定する場合には、以下のコマンドを用います。

stcox 説明変数 ,オプション

前述のとおり、stset コマンドで被説明変数(failvar)を認識させていますので、ここでは被説明

変数を指定する必要はありません。
　先ほどの企業倒産のデータを使って、実際に Cox モデルを推定してみましょう。推定を行うために、以下のようにコマンドを入力します。

stcox slsprofi llabor wage_f

推定の結果は以下のように出力されます。

```
         failure _d:  died
   analysis time _t:  (closeyear-origin)
             origin:  time origin

Iteration 0:   log likelihood = -626.87893
Iteration 1:   log likelihood = -623.83754
Iteration 2:   log likelihood = -603.29044
Iteration 3:   log likelihood = -602.39677
Iteration 4:   log likelihood = -602.36518
Iteration 5:   log likelihood = -602.36512
Refining estimates:
Iteration 0:   log likelihood = -602.36512

Cox regression -- Breslow method for ties

No. of subjects =         2039                     Number of obs   =      2039
No. of failures =           90
Time at risk    =        63310
                                                   LR chi2(3)      =     49.03
Log likelihood  =   -602.36512                     Prob > chi2     =    0.0000

------------------------------------------------------------------------------
          _t |      Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
  slsprofit |  -8.229497   1.207052    -6.82   0.000    -10.59528   -5.863718
     llabor |  -.2330196    .076712    -3.04   0.002    -.3833724   -.0826669
     wage_f |  -.1396185   .0821836    -1.70   0.089    -.3006953    .0214583
------------------------------------------------------------------------------
```

(2)分布を仮定した推定

　ベースライン・ハザードの分布の形を仮定してハザード関数を推定する場合、以下のコマンドを用います。

streg 説明変数 ,dist(分布名)

dist(分布名)の代表的な例として、以下のようなものがあります。

dist(weibull) ：ベースライン・ハザードにワイブル分布を仮定
dist(exponential)： 〃 指数分布を仮定

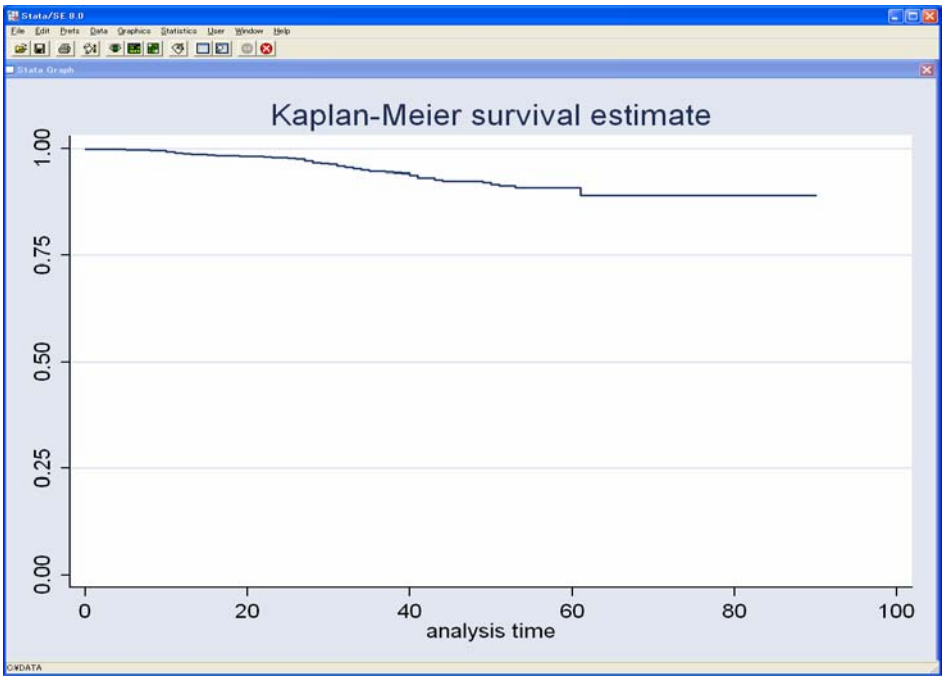
(3)Kaplan-Meier 分析

生存関数(あるいはハザード関数)をノンパラメトリックに推定する方法として、Kaplan-Meier 分析があります。詳しい説明は他の教科書に委ねますが、各期におけるイベントの発生確率を掛け合わせたものを生存関数(Kaplan-Meier 推定量)として、時間の変化とともに生存確率がどのように変化するかを分析する手法です。生存確率と時間の関係(Kaplan-Meier survivor curve)をグラフにする場合、以下のコマンドを用います。

```
sts graph
sts graph, na
```

二行目は、累積のグラフを書く場合に用います。

先ほどの企業倒産のデータを使って Kaplan-Meier survivor curve を書く(sts graph)と以下のように出力されます。



索　　引

P

R

S

T

U

W

X

あ

か

さ

た

は

ま

や

ら