Stata 统计软件使用入门

司继春

上海对外经贸大学统计与信息学院

Stata 是在统计学、计量经济学、社会学、政治学等领域非常常用的统计软件，其名字是英文单词 Statistics and data 的缩写。Stata 不仅提供了统计学与计量经济学最常用的模型，还提供了非常方便的数据管理工具，极大的减轻了统计应用的工作量。除此之外，Stata 还提供了强大的编程功能，用户不仅可以使用 Stata 编写脚本，还可以编写自己的 Stata 命令等。此外，Stata 还自带 Mata 语言——一个类似于 Matlab 的矩阵运算语言，极大的扩展了 Stata 的可编程能力。在这节中我们就简单介绍一下 Stata 的使用方法。

1 Stata 介绍

Stata 为商业软件，因而必须购买才能使用。当然，如果学校或者组织已经购买，可以直接联系学校相关部门索取使用权。一般来说，更新版本的 Stata 提供了更快的速度以及更多的功能，因而推荐尽量使用更高版本的 Stata，截至目前（2017 年 10 月），最高版本为 Stata 15。由于 Stata 14 之后使用了 Unicode 编码，解决了跨系统时的乱码问题，因而推荐尽量使用 Stata 14 之后的版本，避免多人合作时出现乱码问题。

在支持的操作系统方面，Stata 几乎支持市面上所有主流的桌面操作系统，如 Windows、Ubuntu（Linux）、MacOS 等。此外，同一版本号的 Stata 还区分不同的版本，一般分为 IC 版、SE 版和 MP 版本，版本之间的首要区别在于能够处理的数据量的不同，比如 IC 版本最多只能支持 2048 个变量，SE 版本支持 32767 个变量，而 MP 版本多支持多达 120000 个变量。此外，MP 版本还支持多核并行运算，因而大大提高了运算速度。

在安装好 Stata 之后，直接点击 Stata 图标可以进入 Stata 的图形用户界面，在 Linux 中，可以通过运行 Stata 安装目录的 xstata（或者 xstata-se、xstata-mp）打开 Stata 的图形用户界面[1]。

图 (1) 是在 Ubuntu 下 Stata 的界面，主要包括了菜单栏、工具栏、结果（Results）、变量（Variables）、变量属性（Properties）、命令历史（Review）和命令输入框以及最下方显示当前文件夹的状态栏。

[1]在 Linux 中，stata、stata-se、stata-mp 三个程序文件可以用来在终端中打开无图形用户界面的 Stata。

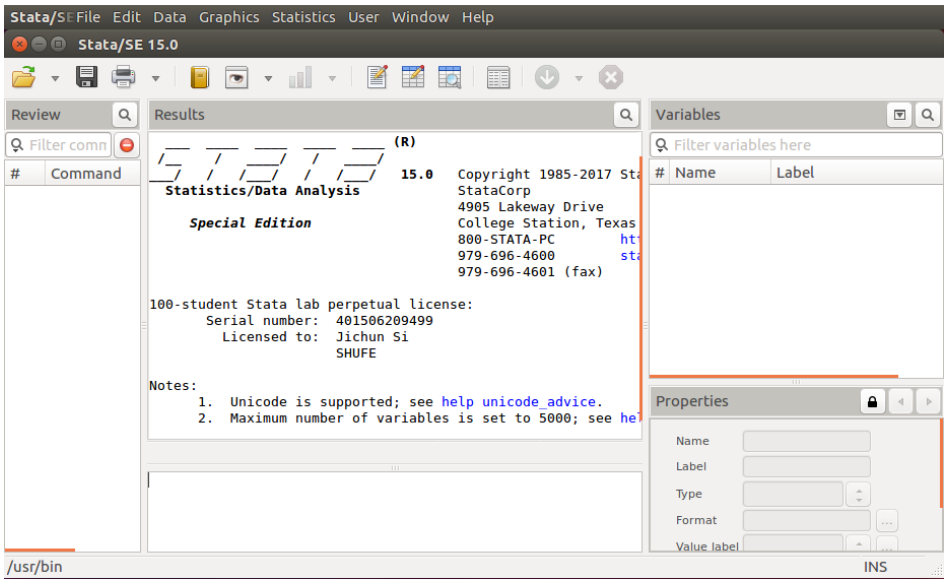

图 1: Stata 窗口

- 菜单栏：包括常见的文件、编辑、窗口和帮助等菜单，此外，还有：数据（Data）菜单，包括了数据整理的常用功能；画图（Graphics）菜单，包括了各种画图功能；统计（Statistics）菜单，包括了 Stata 提供的统计分析功能。
- 工具栏：包括常用的打开、保存、打印等功能，此外，还有日志（Log）、Do-文件编辑器（Do-file Editor）、数据编辑器（Data editor）、数据浏览器（Data brwoser）、和变量管理器。其具体功能我们将在下面介绍。
- 结果（Results）：命令运行的结果将显示在此，右键单击选择 Preferences 可以更改显示风格。
- 变量（Variables）：显示了内存中数据的变量名（Name）及其标签（Label），其中标签是变量的具体描述，我们将在下面介绍。双击某个具体的变量可以快速将该变量名填在命令输入框中。
- 变量属性（Properties）：选中的变量的属性，如数据类型等。
- 命令历史（Review）：在 Stata 中所有执行的命令历史都显示在这里。即使使用菜单等鼠标操作，在执行时也会转化为 Stata 命令执行，同时显示在命令历史中。单机某一条具体的历史可以快速将命令填在命令输入框中，双击会直接运行选中的命令。
- 命令输入框：即光标的位置，由于所有的 Stata 操作都可以通过命令方式完成，因而该输入框在 Stata 的使用中起着至关总要的作用。

尽管多数操作都可以使用菜单和鼠标完成，然而我们仍然强烈推荐使用命令的方式操作 Stata。

在 Stata 中使用命令是非常简单的。比如当我们打开 Stata 时，会提醒我们「Maximum number of variables is set to 5000; see help set_maxvar.」，即变量个数上限被设置为了 5000，可以使用 set maxvar 来修改变量个数的限制，比如可以再命令输入框中使用：

```
set maxvar 10000
```

将变量最大个数设置为 10000。此外，该提醒还告诉我们，如果对于 Stata 命令有任何的疑问，可以使用 help 命令获得帮助，比如：

```
help set
```

命令就可以得到关于 set 命令的所有使用帮助。Stata 的 help 命令为每个命令都提供了非常详尽的解释说明，如果对于某个命令的使用有疑问，可以直接使用「help」得到帮助。此外，如果需要更加详细的命令使用帮助，还可以在 Stata 安装文件夹下的「docs」文件夹下面找到非常完整的 Stata 使用手册[2]，是 Stata 最权威、最全面的使用文档。

从 help set 的结果中我们可以看到，set 命令有非常多的用法。比如在执行某项统计分析之后，很有可能会出现结果太长在结果窗口中一屏幕不能完全显示的情况，此时程序会终止，并显示「—more—」，需要手动点击才能继续程序的运行。如果希望程序不要中断，将所有结果一并显示出来，可以使用 set more off 命令。此外，可以使用 set matsize 设定矩阵的最大的规模，当 Stata 出现「matsize too small」的时候可以使用该命令；在老一点的 Stata 版本中，可能还需要使用 set max_memory 设置 Stata 可使用的最大内存数量。

Stata 具有种类繁多的命令，然而起命令有一定的规律，一般来说，Stata 的命令具有如下格式：

```
[prefix:]command [varlist] [=exp] [if] [in] [weight]
                 [using filename] [, options]
```

其中方括号代表可选项，以上语法可以在「help language」中找到。各个部分含义如下：

- prefix：前缀，为可选项，通常用来对命令进行一些必要的修饰，前缀语句结束后一定有一个冒号以示区分。比如如果在一个命令前面加「quietly:」，那么该命令运行将不会在结果窗口中显示任何结果；而如果加入「noisily:」，将会强制显示结果。

- command：命令的名称，如上面已经介绍的 set 命令、help 命令等，以及用于描述性统计的 summarize、用于回归的 regress 命令等。

[2]其中 i.pdf 提供了详细的索引。

- varlist：需要进行操作的变量列表。
- =exp：表达式，将在下面具体介绍。
- if：某个命令可能不需要对所有的观测进行操作，可以使用 if 子句挑出那些需要被操作的观测。
- in：同样是挑选观测进行操作，只是挑选标准为变量的排序（行号）。比如「br in 1/10」仅仅显示前十个观测。
- weight：权重，在 Stata 中，默认所有的观测都是等权重的，然而由于抽样等原因，每个观测的权重可能并非相等，此时可以使用 weight 子句指明权重，Stata 支持 fwight, pweight, aweight, iweight 等四种权重，具体说明可以 help weight 中查看。
- using filename：指定需要操作的文件名。
- options：命令的选项，跟在一个英文逗号后面，在多数命令中为可选项，控制着命令运行的各种细节。当有多个选项时，只需要一个逗号即可，不同选项之间用空格隔开。

我们将在接下来的学习中不断用到上述命令结构。

此外，除了 Stata 自带的命令意外，还有大量其他非 Stata 自带的、用户编写的命令可以使用。在使用这些命令之前，需要首先安装这些命令。安装命令一般有如下两种方式：

- ssc 安装：直接输入「ssc install 命令名」即可。比如，我们经常会使用 outreg2 命令导出统计分析结果，而这一命令不是 Stata 自带命令，我们可以直接使用「ssc install outreg2」安装这一命令。
- help 安装：直接输入「help 命令名」，Stata 会自动搜索相关的命令，并返回在《Stata Journal》中相关的结果。可以在在弹出的窗口中找到「st****」（* 代表数字），点击进入相应界面，点击「click here to install」安装。

在比较老的版本的 Stata 中，以上 help 命令可能需要换成 findit (search) 命令。

最后，很多 Stata 命令还有缩写的形式，一般为命令名称的前几位字母。比如描述性统计「summarize」命令可以直接使用「su」或者「sum」替代，回归命令「regress」可以用「reg」代替，「browse」可以用「br」代替等等。在命令的 help 中，如果命令（或者选项）有一部分字母加下划线，那么下划线部分就是该命令或者选项的缩写。

2 文件与变量

Stata 有所谓的「工作目录」的概念，设定了工作目录之后，如果接下来不指明路径，那么所有的打开文件、保存文件的操作都默认在这个文件夹中，

或者以这个文件夹为基准。比如使用命令「cd /home/sijichun/」即将工作文件夹修改到「/home/sijichun」目录下，Windows 下类似。此外，Stata 还支持 Linux/Unix/Windows 相似的基本文件命令，比如可以使用「pwd」命令获取当前工作目录，使用「ls」或者「dir」命令显示工作目录的文件和目录，使用「rm」或者「erase」命令删除文件。在接下来的介绍中，我们假设在当前目录下可以找到所需要的所有文件（比如数据文件.dta 或者.do 文件），如果提示文件找不到，请自行使用 cd 命令改变路径。

Stata 数据文件以.dta 作为后缀名，我们可以使用「use」命令打开数据文件，比如，可以使用「use "cfps_adult.dta"」打开数据集「cfps_adult.dta」[3]。尽管多数情况下文件名「cfps_adult.dta」可以不加双引号，然而加双引号是非常良好的习惯。此外，后缀名.dta 也可以省略。

如果在打开数据之前，Stata 内存中已经有数据，那么以上 use 命令会报错。解决以上问题可以有两种方法：

- 使用 clear 命令，清除内存中的所有数据；
- 为 use 命令加入 clear 选项，即：「use cfps_adult, clear」。

注意以上命令将会清除内存中的所有数据，所以在使用以上命令时，如果内存中的数据仍然有用，一定要保存数据，保存数据可以使用 save 命令，比如「save "adult.dta"」，同样，引号和.dta 可以省略。如果文件已经存在，save 命令仍然会报错，需要在 save 命令后面加入 replace 选项。此外，如果使用 use 命令打开某个数据集，进行一定操作之后，希望覆盖原始数据，可以忽略文件名，直接使用「save, replace」即可。当然在实际操作中，由于覆盖是一个不可逆操作，因而建议在任何情况下不要覆盖原始数据。

打开数据文件后，可以通过点击工具栏上的数据浏览器查看数据，或者直接使用「brwose」命令打开数据浏览器查看数据。此外，还可以在 browse 命令后面加 if 或者 in 子句，比如命令「br if provcd14==11」即可查看所有北京市家庭的数据，而命令「br in 1/10」可以查看前 10 行的数据。

在默认条件下，为了保证数据的安全性，数据浏览器中是不可以修改数据的。如果希望单独修改数据，需要在工具栏或者数据浏览器的工具栏中点击数据编辑器进行编辑。或者，可以使用「edit」命令打开数据编辑器。

Stata 还支持从多种数据数据源中导入、导出数据，包括 Excel、CSV、SAS 格式等。可以从「File 菜单-import」中打开导入数据的对话框，选择需要导入的文件名，以及相关的选项即可导入数据。在 Excel 或者文本文件（delimited text data）中，可以选择直接使用数据的第一行作为变量名；在导入文本文件时，可以选择编码（encoding）避免乱码问题。当然，以上所有导入导出操作也可以使用 import/export 命令完成，详细选项可以查看 help。

[3]本讲义中的数据文件可以在https://github.com/sijichun/MathStatsCode中找到，接下来，我们将使用北京大学中国家庭跟踪调查（China Family Panel Studies, CFPS）2014 年部分数据和变量作为示例。

值得一提的是，如果导入的数据在数据浏览器中显示为褐色，说明该列数据被导入成为字符串。如果认为该列变量应该为数值型，可以使用 destring 命令将字符型转化为数值型，具体选项可以查看 help。当然，如果一旦出现这个问题，在使用 destring 之前，检查哪些数据中可能出现了被认为是字符串的字符是非常有必要的。

Stata 中的数据是以「变量（variables）」为基本单位的，一个变量就是数据表中的一列。比如，当我们打开数据「cfps_adult.dta」之后，我们可以在「变量」窗口中看到「pid, fid14, provcd14,...」等变量名，而打开数据浏览器，数据的列名称即为变量名。此外，在变量窗口中每个变量都对应着一个「标签（label）」，用来具体描述变量的含义。

在 Stata 中我们可以使用「generate」命令产生新的变量，该命令基本语法为：

```
gen [type] newvar =exp [if] [in]
```

其中：

- type 为数据类型，为可选项。Stata 支持包括整型（int）、长整型（long）、浮点型（float）、双精度型（double）以及字符串等类型，具体可以使用「help data types」查看 Stata 支持的数据类型的完整说明。
- newvar 为新的变量名，最长为 32 个字符，以字符或者下划线（_）开头，不能以数字开头，不能为保留关键字。实际上由于新版的 Stata 支持 Unicode 编码，因而中文也可以被用作变量名，然而出于可移植性等原因，并不建议使用中文变量名。
- exp 为表达式，通常为其他变量的算数运算或者函数，常见的算数运算符，比如加（+）、减（-）、乘（×）、除（/）、幂运算（^）等都被支持。

比如，以下代码就产生了个人总收入中除工作收入之外的其他收入：

```
use "cfps_adult.dta"
gen other_income=p_income-qg12
```

值得注意的是，在以上 gen 命令运行结束之后，Stata 提示「(61 missing values generated)」，即除正常数据以外，产生了 61 个缺失值（missing values）。在 Stata 中，缺失值以一个英文句号「.」来表示，我们可以使用「br if other_income==.」来查看具体哪些观测其新生成的变量为缺失值。或者，也可以使用：

```
br other_income p_income qg12 if other_income==.
```

只显示 other_income 变量为缺失值的 other_income p_income qg12 三个变量。

在上面的「br」命令中，我们使用了「if」子句。正如前面介绍的，「if」子句用于挑出那些我们希望进行操作的观测，其后面跟着逻辑语句。在 Stata 中，支

持常见的关系运算符如大于（>）、小于（<）、大于等于（>=）、小于等于（<=）、等于（==）、不等于（!=）以及逻辑运算符，如逻辑与（&）、逻辑或（|）、逻辑非（!）。例如，以下命令：

```
gen post90=1 if cfps_birthy>=1990
```

创建一个新的变量 post90，对于 90 后，post90=1，否则为缺失值「.」。而如果想要生成一个变量，比如对于 90 后 post90s=1，否则等于 0，那么可以直接使用以下命令：

```
gen post90s=cfps_birthy>=1990
```

即如果表达式为逻辑语句，那么当某个观测判断为真时 =1，否则 =0。

需要注意的是，在 Stata 中，缺失值「.」是一个比任何实数都大的值，因而如果使用「sort」命令对刚刚生成的 post90 变量进行生序排序：

```
sort post90
br post90
```

在弹出来的数据浏览器中，我们会发现所有的缺失值都排在了后面。因而实际上如果 cfps_birthy 含有缺失值，那么其对应的新产生的 post90/post90s 都等于 1，而非缺失值。这就会导致可能会有一些观测，由于 cfps_birthy 值缺失，而使用以上代码生成 90 后的虚拟变量时，把这些观测误认为是 90 后。因而在产生以上变量之前，一定要先检查 cfps_birthy 值是否有缺失的情况。

除了使用算数运算、逻辑运算作为 gen 命令的表达式之外，Stata 还提供了大量的函数以供使用，比如常见的取绝对值（abs()）、对数函数（log()）、指数函数（exp()）、开平方（sqrt()）、取整函数（int()）、向上取整（ceil()）、向下取整（floor()）、取余函数（mod()）、logit 函数（logit()）等等，此外，还有大量的日期函数、统计函数、三角函数、字符串函数以及随机数生成函数等等，可以使用「help functions」查询 Stata 所支持的函数。

需要注意的是，使用函数也有可能产生缺失值的情况。比如自然对数 log() 其定义域为 $(0, \infty)$，如果对 0 或者小于 0 的数取对数，就会产生缺失值。取对数在数据分析中是非常常见的操作，所以在取对数时一定要注意检查数据的取值范围，按照接下来的描述性统计的做法检查数据是非常有必要的。

最后，Stata 的所有系统和用户定义的标量都可以用在 gen 语句的表达式中。在 Stata 中，支持如下几个系统变量：

- _n：观测的行号
- _N：所有观测的个数（行数）
- _pi：圆周率 π
- _rc：最近一次 capture 命令捕获的错误代码

- _b[varname]：最近估计的统计模型中，变量 varname 的系数
- _se[varname]：最近估计的统计模型中，变量 varname 的标准误

比如，如果我们需要生成一个新的变量，其值为行号，那么我们只需要使用命令：

```
gen id=_n
```

如此便生成了一个新的变量 id，其值为观测在数据浏览器中的行号。其他系统变量作为标量或者字符串，都可以使用「display」命令，比如「di _pi」将显示 3.1415927，而「di _N」将会显示目前内存中的观测数。

以上介绍了生成数据，我们还可以使用「replace」命令来修改数据，其语法与 gen 命令一样。比如，我们可以结合 gen 和 replace 命令生成 90 后的虚拟变量：

```
gen post90=1 if cfps_birthy>=1990
replace post90=0 if post90==.
```

由于第一条 gen 命令对于非 90 后产生了缺失值，因而第二条使用 replace 命令将缺失值修改为 1。以上两条语句与：

```
gen post90s=cfps_birthy>=1990
```

是等价的。同样，以上两种方法都要注意 cfps_birthy 值缺失的问题，即我们可能会错误地将 cfps_birthy 值缺失的观测误认为是 90 后。

作为示例，假设我们希望生成 100 个观测，分别是 50 个个体在 $t = 0, 1$ 两期的对数收入，假设第 0 期的对数收入 $x_{i1} \sim N(10, 5)$，第 1 期的对数收入 $x_{i2} \sim N(11, 6)$，那么生成以上伪数据可以使用如下代码：

```
clear
set obs 100
gen id=ceil(_n/2)
gen time=mod(_n-1,2)
gen x=rnormal()*sqrt(5)+10 if time==0
replace x=rnormal()*sqrt(6)+11 if x==.
```

其中我们首先使用 clear 命令清除所有数据；由于现在内存中没有任何数据，因而我们必须使用 set obs 设定需要产生的随机数的个数，即总的观测数；接下来我们使用 ceil() 向上取整函数，得到了行号除以 2 的向上取整，作为个人的 id，同时使用行号-1 除以 2 的余数作为时间；最后，我们使用了 rnormal() 函数产生标准正态分布，并变换到我们需要的正态分布。

由于变量名的诸多限制，以及为了编写代码时的方便，一般变量名不会是数据的完整描述。特别是在一些调查数据中，变量名通常是被一些代码所替代。

此时我们可以使用变量的「标签 (label)」对变量添加描述。使用「label variable」命令，我们可以使用以下代码为我们生成的 x 添加了解释说明：

```
label variable x "随机生成的对数收入"
```

这样我们就可以在变量列表中看到变量的解释说明了。值得注意的是，标签虽然是对变量的解释说明，但也不宜太长，因为在画图的时候，默认的标题是标签，在没有标签的情况下使用变量名，如果变量的标签太长会导致画图时的坐标轴标题太长。

除了可以对变量名加标签之外，还可以对数据的值加标签。对于分类数据，我们在 Stata 中经常以数值进行存储，比如对于「性别」这个变量，我们经常会使用 0 代表女性，1 代表男性；对于「省份」这个变量，我们经常会使用 11 代表北京，31 代表上海，41 代表河南，37 代表山东等等。尽管使用数字代替分类变量在处理时非常方便，但是不便于人们阅读，为了解决这个问题，可以使用「label」命令为数据加标签，如以下代码为上述生成的 post90 变量添加了标签：

```
gen post90s=cfps_birthy>=1990
label def lab_post90 0 "90前"
label def lab_post90 1 "90后", add
label values post90s lab_post90
```

我们首先使用「label define」命令定义了一个标签，叫做「lab_post90」，并将 0 定义为「90 前」，1 定义为「90 后」，进而使用「label values」命令将刚刚定义的标签「lab_post90」附加给变量「post90s」。如此我们打开数据浏览器，「br post90s」，就可以看到这个变量显示的是蓝色的「90 前」或者「90 后」的标签，而非数字。注意数据的标签显示为蓝色，而字符串为褐色，标签仅仅是为了显示的便利，变量的值仍然是 0 或者 1。可以通过「h la」查看 label 的其他用法。

以上介绍了生成变量，与之相对应的是删除变量或者观测。在 Stata 中，可以使用 drop 和 keep 两个命令完成对变量或者观测的删除。其中「drop varlist」代表删除「varlist」中的变量，而「keep varlist」代表只保留「varlist」中的变量，其他变量全都删除。如果想要删除某些观测而非变量，可以在后面加 if 或者 in 子句，比如「drop if」就删除了满足条件的所有观测，而「keep if」则保留了满足条件的所有观测。比如在 cfps_adult.dta 中使用：

```
drop if te4==-8 | te4==-1
```

就删除了所有最高学历为「不适用」或者「不知道」的观测。当然，以上语句等价于：

```
keep if te>0 & te4!=.
```

即只保留 te4 为正值且不缺失的观测。

最后，我们还可以使用「varname[n]」来单独读取变量「varname」的第 n 个观测的值，比如「x[1]」就是变量 x 的第一个观测，而「x[_N]」就是变量 x 的最后一个观测。

以上特性经常被用来生成滞后项。在 Stata 中，我们可以通过「tsset」命令设定时间序列，其语法为：

```
tsset timevar
```

其中 timevar 为代表时间的变量，在数据中必须是唯一的[4]。在定义了时间变量之后，我们就可以使用所谓的滞后和向前算子（L. 和 F.）来生成滞后变量，以下程序展示了使用两种方法产生滞后和向前变量：

```
clear
set obs 100
gen t=_n
gen x=rnormal()
drop if t==50
gen x_lag_2=x[_n-1]
gen x_fwd_2=x[_n+1]
gen x_lag2_2=x[_n-2]
tsset t
gen x_lag=L.x
gen x_fwd=F.x
gen x_lag2=L2.x
order t x x_lag_2 x_lag x_fwd_2 x_fwd x_lag2_2 x_lag2
```

首先我们产生了一个唯一可识别的变量 t，以及一个随机的 x。接下来我们使用了两种方法产生 x 的滞后和向前变量，分别通过方括号实现和滞后、向前算子实现，最后使用 order 命令改变这些变量的显示顺序方便比较。对于 t=1 的观测，由于其滞后项为 t=0 或者 _n=0，在数据集中没有 t=0 或者 _n=0 的观测，因而不管用什么方法，产生的 x_lag(_2) 的第一个观测都是缺失值，x_fwd(_2) 的最后一个观测同理也为缺失值。

值得注意的是，尽管以上两种方法产生的滞后和向前变量在大多数情况下都是相等的，但是注意到在以上程序中，我们删除了 t=50 的观测。如果使用第一种方法，即方括号的方法，其产生新变量完全根据行号（_n），删除 t=50 的观测，那么第 50 行就变成了 t=51 的观测，其 _n-1 为 t=49 的观测，因而 x_lag_2 在第 50 行的值（t=51）就是 x 第 49 行的值；而如果使用第二种方法，由于 t=51 的滞后为 t=50，但是我们删除了 t=50 的观测，所以 x_lag 在第 50 行的值（t=51）为缺失值。

[4] 与之相对应的还有 xtset，即定义面板数据，在面板数据中要求每个个体内部时间变量是唯一的。

3 描述性统计

描述性统计是进行数据分析所必须的先行步骤，包括使用各种描述性统计量制作的表、图等对数据进行探索性的分析。Stata 提供了丰富的描述性统计工具，我们将简要介绍一下使用 Stata 做描述性统计最常用的工具。

在 Stata 中做描述性统计最常用的命令就是「summarize」，该命令提供了多数常用的描述性统计量，比如最常用的观测数（Obs）、均值（Mean）、标准差（Std. Dev.）、最小值（Min）、最大值（Max）等。除此之外，在 su 命令后面加上「detail」选项，可以汇报更加详细的描述性统计量，比如一些重要的分位数、偏度、峰度等等。

如果 su 命令后面不加任何变量，那么默认会汇报内存中所有数值型变量的描述性统计。或者，在 su 后面也可以加需要进行描述性统计的变量列表。标量列表可以一次性把要做描述性统计的所有变量都列举出来，或者可以使用通配符。正如多数系统的命令一样，符号「*」可以作为通配符，比如「qg*」代表所有以 qg 开头的变量；此外，Stata 还支持「-」通配符，即按照变量的排列顺序，从某个变量到另外一个变量的所有变量，比如「qg12-p_income」即代表从 qg6 到 p_income 的所有变量。在文件 cfps_adult.dta 中，以下三条命令是等价的：

```
use "cfps_adult.dta", clear
su qg12 qg1203 p_income
su qg12* p_income
su qg12-p_income
```

在论文中我们经常需要汇报描述性统计或者其他统计分析的结果，如果手动一个个输入到 Word 或者 LaTeX 中，是一件非常痛苦的事情。在 Stata 中，我们可以非常方便的使用「outreg2」命令导出我们的统计结果。注意该命令不是 Stata 自带的命令，因而需要使用「ssc install outreg2」进行安装。其语法为：

```
outreg2 using myfile, [{sum(log)|sum(detail)} replace
        eqkeep() eqdrop() keep() drop()]
```

其中：

- myfile 为要导出的文件名，该命令会通过后缀名（.doc 或者.tex）判断需要导出的格式；
- sum(log) 或者 sum(detail) 为指定要导出的描述性统计是否要报告更详细的统计量。为了做描述性统计，两者必须选择一个；
- replace 表示，如果 myfile 存在，那么覆盖这个文件；
- eqkeep() 和 eqdrop()，表示要保留或者舍弃的描述性统计量；
- keep() 和 drop()，表示要保留或者舍弃的变量。

VARIABLES	(1) N	(2) mean	(3) sd	(4) min	(5) max
qg12	37,147	8,415	18,816	-8	800,000
qg1203	37,147	1,649	18,023	-8	3.000e+06
p_income	37,086	8,934	18,819	-9	442,000

表 1: outreg2 示例

在运行「outreg2, sum()」命令之前，不需要先做运行 su 命令。比如如下命令：

```
use "cfps_adult.dta", clear
outreg2 using summary.tex, replace sum(log)
        keep(qg12-p_income)
```

会得到如表 (1) 的结果，一个标准的描述性统计表。

注意到，在表 (1) 中，三个变量均出现了负数，然而根据三个变量的标签，这三个变量分别代表三种收入，但是收入最小值应该为 0，不可能出现负数。这就是我们做描述性统计的一个重要原因，即发现数据中的异常，并及时处理这些异常。通过查看数据的标签以及调查问卷，我们发现这些负数可能代表了缺失、没有调查等不正常的情况，因而在实际分析中，我们应该对这些情况做出恰当处理，比如最简单的，drop if p_income<0。

此外，为了变成方便，在 Stata 命令运行结束之后，经常会保存一些结果在内存中。这些结果通常分为 r-class 和 e-class[5]。其中 r-class 是一般的返回结果，而 e-class 专指估计（estimate）的结果。这些结果可能是一些标量，也有可能是一些矩阵等等。

对于 su 命令，可是使用 help su 命令查看 su 命令结束后保存在内存中的结果。比如，在 su 命令结束后，r(N) 就保存了观测的个数，r(mean) 保存了均值，而 r(sd) 保存了标准差等等。如果在运行 su 命令时指定了不止一个变量，那么这些 r-class 结果只会保存最后一个变量的结果。

我们可以方便的在接下来的命令中使用这些返回变量。比如下面的代码就产生了对 p_income 的标准化后的变量：

```
use "cfps_adult.dta", clear
drop if p_income<0
quietly: su p_income
gen std_p_income=(p_income-r(mean))/r(sd)
su p_income std_p_income
```

由于我们只是希望计算均值，因而使用了 quietly 禁止 su 命令的输出；产生了标准化的收入之后，我们再次做描述性统计，发现新生成的变量已经被标准化

[5]还有其他类型的返回结果，如 s-class、n-class 等，详见 help return。

为均值为 0、方差为 1 的变量了。

以上 su 命令给出的描述性统计，如均值、方差等，只对于连续性随机变量，或者 0-1 型随机变量有意义，然而有一些变量属于分类变量或者顺序变量，它们的值代表了不同的水平（level），其绝对数值的加减等算数运算没有意义，因而均值、方差等描述性统计量也就没有意义了。

比如，「最高学历」这个变量主要有文盲、小学、初中、高中等不同水平，在数据中分别用 1\2\3\4 来代替，但是这些数字只是代表了不同水平，其算数运算没有意义。

处理这类数据我们经常会画频率或者频数表。在 Stata 中，我们可以使用「tabulate」命令制作频率表。比如，如果我们希望制作最高学历这一变量的频率表，那么只要使用命令「ta te4」即可。注意在结果表格里面，其分类是使用的数据的 label 而不是具体的值，如果需要使用具体的值放在表格里面而不用 label，可以在命令后面加「nolabel」选项。

同样的，在结果表格中，我们发现有负值存在，比如-8 代表「不适用」，即没有调查这个问题，「-1」代表不知道。在接下来更详细的统计分析中一定要注意对这些观测做恰当处理。

当我们有两个分类变量需要同时分析时，通常会使用所谓的列联表。比如，我们希望同时研究最高学历和性别这两个变量，研究最高学历是否与性别有关系，在 Stata 中可以直接使用 tab 命令完成，比如：

```
use "cfps_adult.dta", clear
drop if te4<0
tab te4 cfps_gender, ch
```

我们首先删除了那些学历不正常的数据，然后使用 tab 命令只做了列链表，如表 (2) 所示。此外，为了对这两个变量的独立性做假设检验，我们还添加了 ch 选项，汇报 Pearson's χ^2 统计量。

表中我们可以看到两个分类变量所组成的各个分类里面的频数，以及边际频数。最后，Pearson's χ^2 检验 $p-value$ 接近于 0，因而可以拒绝原假设，认为最高学历和性别不是独立的。

此外，分类变量还与 Stata 中的「因子变量」有着密不可分的联系。对于一个分类变量，我们可以用 tab 命令产生其虚拟变量，比如对于最高学历 te4 这个变量，可以使用：

```
use "cfps_adult.dta", clear
drop if te4<0
tab te4, gen(edu)
```

如此就产生了 7 个虚拟变量（dummy variables）edu1-edu7，分别对应着 te4 的 7 种可能的取值：文盲/半文盲、小学、初中等等。比如，变量 edu1 的 label 为「te4== 文盲/半文盲」，即这个变量对于文盲/半文盲来说 =1，否则 =0；变量

最高学历	性别 女	男	Total
文盲/半文盲	198	134	332
小学	262	388	650
初中	577	708	1,285
高中/中专/技校/职高	250	296	546
大专	126	107	233
大学本科	82	106	188
硕士	4	5	9
Total	1,499	1,744	3,243

Pearson chi2(6) = 40.4383 Pr = 0.000

表 2: 列联表示例

edu2 的 label 为「te4== 小学」，即这个变量对于小学文化的观测来说 =1，否则 =0，后面以此类推。

如果我们使用 su edu* 命令，就可以得到这 7 个虚拟变量的描述性统计，每个变量的均值实际上就是对应种类占全部样本的比例。比如，edu1 的均值为 0.1029，意味着样本中有 10.29% 的人是文盲/半文盲。注意由于所有的 edu* 都是虚拟变量，即 0-1 变量，因而其标准误 r(sd) 总是等于 $\sqrt{r(mean)\cdot[1-r(mean)]}$。

以上我们使用 tab 命令产生了一个分类变量的虚拟变量，而在 Stata 中，可以不用具体产生这些虚拟变量，而是直接用所谓「因子变量」即可进行分析。比如以上的描述性统计我们可以直接用：

```
use "cfps_adult.dta", clear
drop if te4<0
su i.te4
```

命令即可得到虚拟变量的描述性统计，使用 i.te4 与先生成 te4 的虚拟变量，再使用虚拟变量是等价的，不过 i.te4 的好处是避免了产生新的虚拟变量，即麻烦又占内存。这里需要注意两点，首先是，作为因子变量的取值不能为负，因而我们必须首先把 te4<0 的观测删掉，不然会报错；其次是，仔细观察使用因子变量所产生的描述性统计，只汇报了 6 个分类的描述性统计，文盲/半文盲类被忽略掉了，这是由于因子变量经常用在回归分析中，而在回归分析中由于识别条件，因而不能把所有的虚拟变量同时加进去。当使用 i.te4 这类语法时，Stata 会把值最小的分类（文盲/半文盲）省略，作为比较的基准。如果想要改变这个基准的类别，可以使用 ib#.te4，其中 # 为想要使用第几类作为基准，比如如果想要用小学作为基准，那么可以使用 ib2.te4。如果想要全部显示，可以在 su 的选项中添加 allbaselevel 即可。

此外，还可以使用 # 符号将两个或者多个分类变量联合起来产生新的分类变量，即两个分类变量的乘积。比如「i.te4#i.cfps_gender」即产生了两种性别

和七种学历共 14 种组合。如果我们使用：

```
use "cfps_adult.dta", clear
drop if te4<0
su i.te4#i.cfps_gender
```

就得到了 13 组均值和标准差。

除了 i. 算子之外，还有 c. 算子。i. 算子代表后面的变量是分类变量，而 c. 算子表明后面是连续性变量。同样可以使用 # 符号产生新的交互项，比如 i.te4#c.p_income 就产生了 7 个新的变量，即 7 个虚拟变量分别与 p_income 的乘积，比如新产生的第一个变量为：

$$i.te4\#c.p\_income\,(1) = \begin{cases} p\_income & \text{文盲/半文盲} \\ 0 & \text{其他} \end{cases}$$

其他 7 个变量以此类推。使用：

```
use "cfps_adult.dta", clear
drop if te4<0
su i.te4#c.p_income
```

可以查看这七个变量的描述性统计。注意这七个变量都是混合型随机变量，即有离散的部分（te4 是否属于某一组），也有连续的部分（如果属于这一组 p_income 的值）。

除了以上介绍的一些描述性统计的命令之外，还有其他的描述性统计命令可以使用，如相关系数矩阵「correlate」命令、Spearman 秩相关命令「spearman」等等。可以自行查阅相关帮助学习。

Stata 有非常强大的绘图功能。在「Graphics」菜单中可以找到大多数需要的统计图的类型。最常用的图形比如：

- 柱状图（bar）：使用「graph bar」命令，可以对不同变量的描述性统计画柱状图，也可以根据一个分类变量画频率图。
- 直方图（histogram）：使用「histogram」命令，适用于连续的随机变量，是对密度函数的样本近似。
- 线图（line）：表示一个变量与另外一个变量之间的关系，比如可以使 GDP 随着时间的发展趋势，或者消费随着收入的变化等等。
- 散点图（scatter）：同样表示两个变量之间的关系，不同的是将数据点按照两个变量的坐标直接画在图上。
- 箱线图（box plot）：将数据的中位数、上下四分位数、最大值、最小值用箱线表示的图。

- ……

对于这些命令我们将不再具体展开，我们下面将以实际代码展示如何画图。

以上的这些图都可以再次进行组合，即多种图形组合在一张图上。我们可以使用「twoway」命令很方便的将不同的图组织在一张图中，下面我们将通过示例展示 twoway 命令的用法。

此外，还有一些辅助指令，比如可以使用「graph save」命令将生成的图片保存成.gph 格式；使用 graph combine 命令将已经保存的图排列在一张途中；使用 graph export 命令导出.png 等格式的图片文件。

以下是一个画图的综合示例：

```
use "cfps_adult.dta", clear
drop if te4<0
drop if p_income<=0

gen lincome=log(p_income)
quietly: su lincome
gen std_lincome=(lincome-r(mean))/r(sd)
gen norm_dens=normalden(std_lincome)
label variable norm_dens "Std. Normal density"
sort std_lincome
twoway (hist std_lincome)
       (line norm_dens std_lincome)
       , saving(g1, replace)

sort std_lincome
gen empirical_cdf=_n/_N
gen norm_cdf=normal(std_lincome)
label variable norm_dens "Std. Normal CDF"
twoway (line empirical_cdf std_lincome)
       (line norm_cdf std_lincome)
       , saving(g2, replace)

gen age=2014-cfps_birthy
gen age2=age^2
quietly: reg lincome age age2
predict predict_lincome
sort age
twoway (scatter lincome age)
       (line predict_lincome age)
       , saving(g3, replace)
```

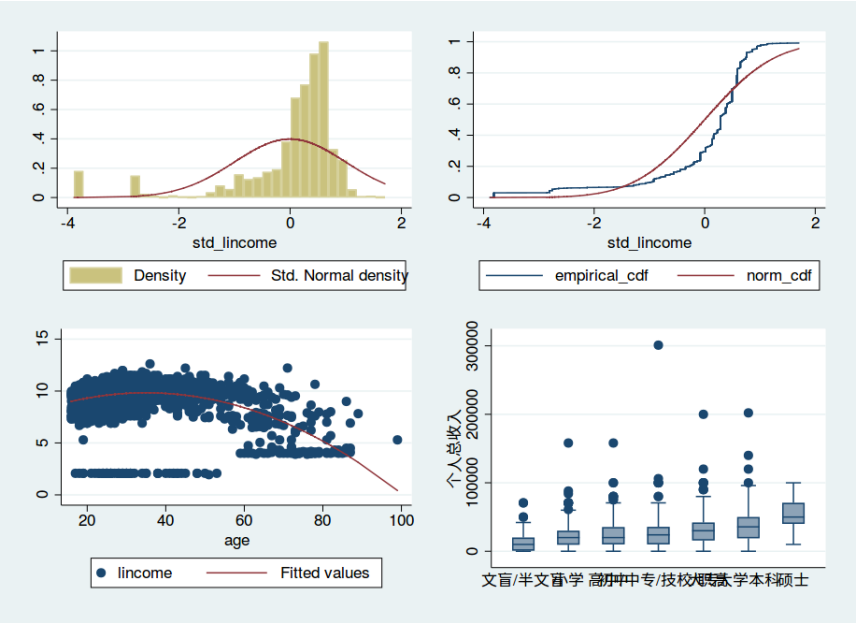

图 2: 画图示例

```
31
32 graph box p_income, over(te4) saving(g4, replace)
33
34 graph combine g1.gph g2.gph g3.gph g4.gph, col(2)
35 graph export example_graph.png, replace
```

图 (2) 给出了以上程序的运行结果。

在以上程序中，我们首先剔除了收入和教育水平存在问题的观测。在第一幅图中，我们首先对个人收入取 log，再将对数收入做标准化处理，得到 std_lincome。接下来，我们使用 normalden() 函数对于每个 std_lincome 都产生其对应的正态分布密度函数值，并对 std_lincome 进行排序，最后画出 std_lincome 的直方图以及标准正态分布的密度函数，并保存为 g1.gph。

在第二幅图中，我们先对 std_lincome 排序，这样 _n/_N 就是其经验分布函数，同时我们用 normal() 函数计算了标准正态分布的分布函数，并将其画在同一张图中进行比较，最后将图保存为 g2.gph。

在第三张图中，我们先生成了年龄和年龄的平方两个变量，并用线性回归模型，用年龄和年龄的平方解释对数收入，使用 predict 命令得到预测值。接着由于我们的横轴是 age，因而需要对 age 进行排序，将散点图和预测值画在一张图中，保存为 g3.gph。

最后，我们根据最高学历进行分类，分别画出了每种学历的对数收入的箱线图，保存为 g4.gph。

得到了以上四张图以后，我们用 graph combine 命令将四张图组合在一起，

并指定四张图排成两列。最终，使用 graph export 命令把最终的图保存成.png 格式。

在 Stata 中有种类繁复的画图命令，每种命令又有大量选项，比如有控制线条、散点、坐标轴、标签、背景的样式、颜色等等等等的各种选项，具体命令和选项可以借助 help 文档进行学习和使用。

4 do-文件与 log-文件

以上我们初步学习了 Stata 命令的一些初步知识，通过在 Stata 窗口的命令输入框中输入命令可以完成 Stata 大部分的操作。然而以上单条命令运行的方式并不方便，比如如果我们有大批量的任务需要完成，如果一次次输入命令，如果一次命令输错了，而 Stata 没有撤销的功能，我们就必须从头再重新来过一次。所以多数情况下，我们使用 Stata 命令都是通过写 do 文件的方式来完成的。

一个 do 文件就是一个以.do 为后缀名的文本文件。我们把需要进行的操作全部书写到一个文本文件里面，并保存成.do 的后缀名，然后直接在 Stata 命令框中输入「do do_filename」就可以批量运行 do 文件里面的所有程序了。或者，在 Linux 命令行下，使用「stata do_filename」（或者是 stata-se 等）可以直接在终端中运行 do-file。

Stata 自带了一个 do 文件的编辑器，可以通过工具栏上的 do 文件编辑器按钮打开。在这个编辑器里面可以直接写 Stata 命令，点击右上方的 Execute(do) 按钮可以一键运行该文件。或者，也可以选中文件的一部分，那么 Stata 将会只运行选中的部分。

我们之前介绍了可以使用「set more off」将–more–关闭，一般我们在写.do 文件时总是会先使用该命令，防止程序因为–more–的出现而终止。

就像所有语言一样，do 文件也有注释。Stata 的注释同样分为单行注释和多行注释。单行注释有两种，一种以两个斜杠开头，即以「//」开头，可以出现在任何位置，只要出现了两个斜杠，就代表从这里到这一行的末尾都是注释。或者，如果这一行都是注释，可以直接以星号「*」开头。注意「//」可以允许注释从一行的后半段开始，而「*」开头的注释必须整行都是注释。多行注释以「/*」开始，以「*/」结束。被注释的部分在 do 文件编辑器中都会显示绿色。

注释有一个巧妙的应用是在 do 文件中换行。在 Stata 中，默认 do 文件每一行是一条命令。但是有的时候命令太长，全都写在一行不便于阅读，一个可行的方案是在命令的中间加三个斜杠「///」后面跟一个回车，然后在新的一行中继续写命令，这样 Stata 就不会认为是新增加了一行。或者，也可以将回车包在多行注释里面，即在「/*」和「*/」中间加入回车，Stata 也会忽略这个回车。

由于经常会使用 Stata 处理变量非常多、非常复杂的调查数据等等，因而养成包括为变量取容易理解的名字、为每一块程序写注释、为关键的行写注释等好的习惯是非常重要的，特别是处理大型任务时，良好的习惯可以大大提高效率。

有的时候我们在 do 文件中不希望某一块代码的输出打印到结果对话框中。

前面我们介绍了使用 quietly: 前缀关闭某一行命令的输出，而在写 do 文件的时候，如果一大块代码都不希望有任何输出，每行都写一个 quietly: 前缀是非常麻烦的。遇到这种情况，可以使用 quietly{} 语句，将所有的不希望输出的语句都写在大括号里面。如果在这个大括号里面有某行语句我们希望其输出结果，就单独在这一行里面加入 noisily: 前缀。以下的程序就关掉了所有的输出，只保留了一个 su 的输出：

```
set more off
use "cfps_adult.dta", clear
quietly{
    su p_income
    gen demean_income=p_income-r(mean)
    noisily: su demean_income
}
```

do 文件不仅仅提供了批量处理的方便途径，也是我们接下来要学习的 Stata 编程的基础。

在通常情况下，Stata 的命令输出结果仅仅在 Reuslts 窗口显示。如果我们希望长久的保留下命令运行的过程（而不是仅仅保留统计结果，比如使用 outreg2），那么可以使用 Stata 的日志文件（log 文件）。

在 Stata 中，可以使用如下命令开始一个新的 log 文件：

```
log using filename [, append replace {text|smcl}]
```

其中 filename 是 log 文件的文件名，append 表明当 filename 存在时，新的内容将会附加在已经存在的文件后面，而 replace 意味着直接覆盖已经存在的文件。Stata 的日志文件有两种格式，文本文件即 text 格式和 smcl 格式。文本文件可以使用任何文本编辑器打开，而 smcl 格式只能用 Stata 打开。如果 log 文件需要传阅给其他人而又不能确保其他人安装了 Stata，可以使用文本文件，否则尽量使用 smcl 格式，因为保留了字体、颜色等格式。

使用以上命令滞后，一直到「log close」命令，中间的命令过程都将被记录下来。如果希望暂时停止 log 文件的记录，可以使用「log on/off」命令临时打开或者关闭 log 文件的记录。更多关于 log 文件的命令和说明可以查看帮助文档，help log。

5 标量、矩阵与宏

levelsof

6 控制流

条件

循环（foreach forvalue）

7 数据处理

变量相关 rename label variable keep drop　　drop if egen
合并数据：merge append
排序：sort
分组：by 前缀 _n、_N 的用法
collapse
其他：reshape：转置数据 duplicates：显示、删除重复观测

8 程序

9 模拟

10 高级编程