

STATA 统计分析教程

目录

STATA统计分析教程	1
目录	2
1 STATA入门	6
1.1 安装	6
1.2 启用和退出	6
1.3 打开和查看数据	8
1.4 寻求帮助与网络资源	9
1.5 命令示例	10
1.6 几个环境设置	11
1.7 复习和练习	12
1.8 附录	13
2 命令语句	15
2.1 掌握命令语句的格式	15
2.2 命令command	15
2.3 变量varlist	15
2.4 分类操作by varlist	16
2.5 赋值及运算=exp	16
2.6 条件表达式if exp	17
2.7 范围筛选in range	17
2.8 加权weight	17
2.9 其他可选项,options	18
2.10 复习与练习	19
3 数据	20
3.1 打开示例数据和网络数据: use	20
3.2 数据类型	21
3.3 数据类型转化	24
3.4 数据显示格式:format	26
3.5 在STATA中直接录入数据: input	27
3.6 导入其他格式数据: insheet	30
3.7 标签数据:label	32
3.8 复习与练习	35
4 数据整理	36
4.1 拆分与连接数据文件要掌握的命令	36
4.2 案例:拆分与连接数据	36
4.3 案例: 连接数据文件	39
4.4 数据重整	39
4.5 案例:数据转置	41
4.6 复习与作业	42
5 函数与运算符	44
5.1 运算符exp	44
5.2 函数概览function	46
5.3 数学函数math functions	47

5.4 字符函数string functions.....	50
5.5 分类操作by.....	51
6 程序	54
6.1 标准的程序文件格式.....	54
6.2 创造自己的命令：与STATA互致问候.....	54
6.3 暂元Macros: local/global.....	57
6.4 自带命令参数.....	59
6.5 scalar标量.....	60
6.6 临时变量和临时数据文件:tempvar和tempfile.....	61
6.7 基尼系数命令的创建案例（选学内容）.....	62
7 流程语句	67
7.1 循环语句:while.....	67
7.3 循环语句:forvalues.....	68
7.3 循环语句:foreach.....	69
7.4 嵌套循环.....	71
7.5 条件语句.....	72
7.6 复习和练习.....	74
8 矩阵	75
8.1 生成矩阵.....	75
8.2 矩阵四则运算.....	76
8.3 矩阵函数.....	78
8.4 随机向量与矩阵代数（选学内容）.....	81
9 绘图	84
9.1 绘图命令.....	84
9.2 几种常用的图.....	89
9.3 同时做多个图by(varname).....	96
9.4 模板及图文件处理.....	98
9.5 附录.....	99
10 随机模拟	100
10.1 伪随机数.....	100
10.2 简单模拟.....	101
10.3 复杂模拟.....	103
10.4 多阶段模拟.....	105
10.5 商店案例.....	107
10.6 练习.....	108
10.7 附录.....	109
11 分布函数	115
11.1 二项分布.....	115
11.2 标准正态分布函数.....	115
11.3 正态分布函数及其反函数.....	116
11.4 服从正态分布的随机数.....	117
11.5 正态分布密度函数.....	118
11.6 分位数.....	119
11.7 卡方分布.....	120

11.8 t分布的分位数.....	122
11.9 F分布	122
12 抽样分布	125
12.1 经验分布.....	125
12.2 均值的抽样分布：正态总体的小样本抽样分布.....	126
12.3 中心极限定理：非正态总体大样本下均值的抽样分布	126
12.4 卡方分布与样本标准差的抽样分布	128
12.5 构造F分布	129
12.6 t分布：未知总体方差时的抽样分布	130
12.7 多元正态分布.....	131
13 参数估计与假设检验	133
13.1 极大似然估计的原理.....	133
13.2 正态总体均值和方差的极大似然估计	133
13.3 最小二乘估计OLS原理	134
13.4 矩估计MM原理.....	135
13.5 区间估计原理.....	135
13.6 假设检验原理.....	136
14 简单回归原理	138
14.1 回归分析原理.....	138
14.2 模拟实验.....	142
14.3 回归报告结果中各项的手工计算.....	143
14.3 线性模型的最大似然估计	145
15 异方差模拟	147
15.1 条件分布图示.....	147
15.2 异方差的后果.....	148
15.3 图形检验与怀特检验.....	150
15.4 检验的功效(选读内容)	151
15.5 估计方法：WLS与GLS.....	154
15.6 广义最小二乘估计与FGLS	155
Equation Chapter 1 Section 1 16 随机过程模拟	157
16.1 时间数据函数.....	157
16.2 模拟白噪声及检验白噪声	158
16.3 模拟自回归过程AR并检验稳定性	160
16.4 模拟移动平均过程MA	163
16.5 序列相关性检验.....	167
16.6 单位根检验.....	168
16.7 平滑分析.....	170
17 计量经济学基本理论模拟	172
17.1 经典假设满足时OLS估计量的小样本性质	172
17.2 条件误差服从正态分布的假设不成立时OLS的小样本性质.....	173
17.3 条件误差服从正态分布假设不成立时OLS的大样本性质	173
17.4 第一假设不成立时	175
17.5 第二假设不成立时	176
17.6 第三假设不成立时.....	177

17.7 第四假设不成立时	177
17.8 第五假设不成立时（略）	177
18 计量经济学综合案例	179
18.1 简单回归分析	179
18.2 多元回归分析	181
18.3 非线性回归分析	182
18.4 回归模型的有效性	184
18.5 实验与自然实验	187
参考文献	189

I STATA 入门

Stata 统计软件包是目前世界上最著名的统计软件之一，与 SAS、SPSS 一起被并称为三大权威软件。它广泛的应用于经济、教育、人口、政治学、社会学、医学、药学、工矿、农林等学科领域，同时具有数据管理软件、统计分析软件、绘图软件、矩阵计算软件和程序语言的特点，几乎可以完成全部复杂的统计分析工作。其功能非常强大且操作简单、使用灵活、易学易用、运行速度极快，在许多方面别具一格。

Stata 的命令语句极为简洁明快，而且在统计分析命令的设置上又非常有条理，它将相同类型的统计模型均归在同一个命令族下，而不同命令族又可以使用相同功能的选项，这使得用户学习时极易上手。Stata 语句在简洁的同时又拥有着极高的灵活性，用户可以充分发挥自己的聪明才智，熟练应用各种技巧，真正做到随心所欲。尽管它也提供了窗口菜单式的操作方式，但强烈建议大家坚持使用命令行 / 程序操作方式，很快你就会体会到使用程序和命令方式所带来的那种随心所欲地处理和分析数据的快感。

Stata 的另一个特点是他的许多高级统计模块均是编程人员用宏语言写成的程序文件（ADO 文件），这些文件可以自行修改、添加和下载。用户可随时到 Stata 网站寻找并下载最新的升级文件。这一特点使得 STATA 始终处于统计分析方法发展的最前沿，用户几乎总是能很快找到最新统计算法的 Stata 程序版本，而这也使得 Stata 自身成了几大统计软件中升级最多、最频繁的一个。

STATA 由美国计算机资源中心(Computer Resource Center)研制，现为 STATA 公司的产品。从 1985 至 2007 的二十多年时间里，已连续推出 1.1, 1.2, ..., 7.0, 8.0, 9.0, 10.0 等多个版本。我们将要学习的是 9.2 版本。

1.1 安装

(1) <http://www.pinggu.org/bbs/dispbbs.asp?boardID=67&ID=97705&page=2> 上有 stata9.rar 下载，但是做正式的论文或工作还是应该尽量用正版软件。

(2) 将其解压到 D:/stata9。

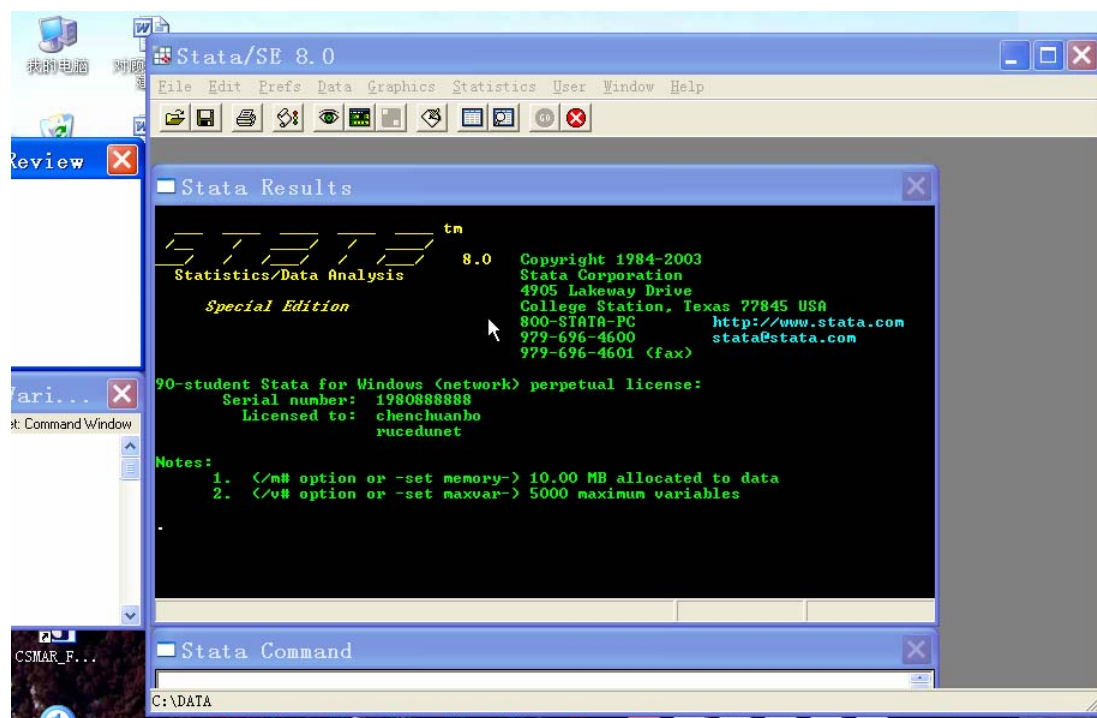
(3) 点击 setup 安装>>改变安装路径到 D:/stata9>>选择 Stata/SE 版本。

1.2 启用和退出

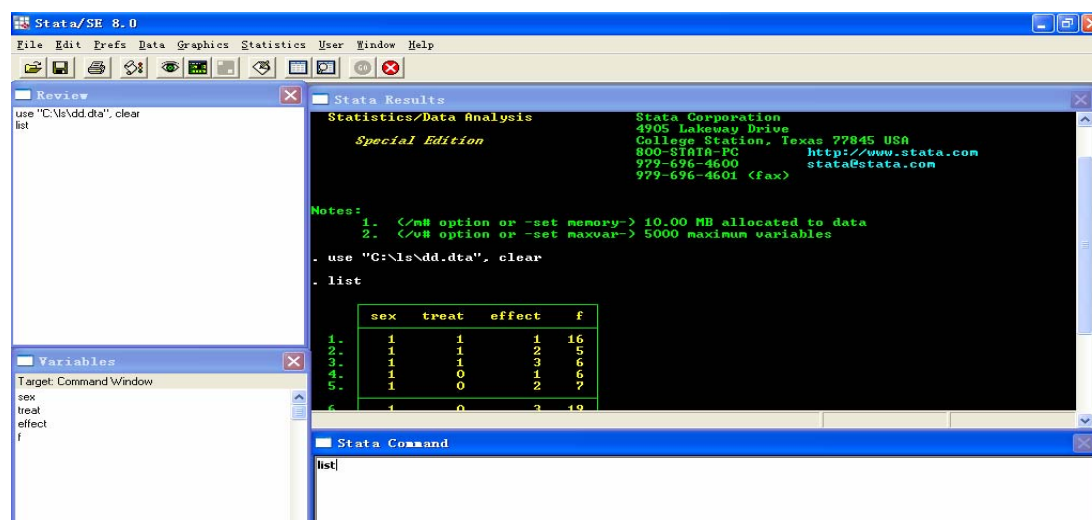
(1) 程序→Stata，即可进入 Stata，启动后出现文件对话框，要求输入注册单位和密码等。



(2) 打开 D:/stata9>> 点击 wsestata>> 打开 sn 文件找到注册信息, 进行注册 (注意用户名和单位要多于 5 个英文字符)。完成注册后, 出现如下画面。



(3) 调整和保存界面设置: 拖动各个窗口, 将其调整为如下合适的格式, 然后选择 Prefs>> save windowing preference



Stata 的界面主要是由四个窗口构成：

结果窗口：位于界面右上部，软件运行中的所有信息，如所执行的命令、执行结果和出错信息等均在这里列出。窗口中会使用不同的颜色区分不同的文本，如白色表示命令，红色表示错误信息。

命令窗口：位于结果窗口下方，相当于 DOS 软件中的命令行，此处用于键入需要执行的命令，回车后即开始执行，相应的结果则会在结果窗口中显示出来。

命令回顾窗口：即 review 窗口，位于界面左上方，所有执行过的命令会依次在该窗口中列出，选中某一行单击后命令即被自动拷贝到命令窗口中；如果需要重复执行，用鼠标双击相应的命令行即可。

变量名窗口：位于界面左下方，列出当前数据集中的所有变量名称。

除以上四个默认打开的窗口外，在 Stata 中还有数据编辑窗口、程序文件编辑窗口、帮助窗口、绘图窗口、Log 窗口等，如果需要使用，可以用 Window 或 Help 菜单将其打开。


(4) 点击右上角的 X 号退出。


建议安装路径为：D: /stata9。一般不要安装在 C 盘下，更不要直接放在桌面上。这是因为我们通常会将数据和程序存储于安装目录下，如果安装 c 盘，一旦计算机出现意外故障，很可能导致我们存储在上面的数据无法恢复。

1.3 打开和查看数据

打开和查看一个数据文件有三种方式，这三种方式分别是窗口式操作、命令式和程序式操作。例：我们要打开 STATA 自带的示例数据文件 `auto.dta`。

1.3.1 窗口执行方式

(1) 点左上角的第一个按钮 ，弹出一个对话框，选择 STATA 软件自带的示例数据文件 `auto.dta`，双击即打开该文件。

(2) 然后点击倒数第四个按钮  图标，弹出一个数据库窗口，显示的是 `auto` 数据文件包含的具体内容。

该数据集共有 12 列 74 行，每一列为一个变量，如第一列为汽车品牌，第二列为价格等；每一行为一辆汽车的相关信息，如第一行的汽车是 AMC Concord，价格为 4099 美元。

(3) 点右上角的 X 号，退出数据窗口。

1.3.2 命令互动执行方式

注意到执行上述操作后，结果窗口新出现了两行白色字体显示的如下内容：

```
use "D:\Stata9\auto.dta", clear
```

```
edit
```

其中，前者为打开 auto 数据文件的命令，后者为查看该数据的命令。将该行文字选中，点右键并选择复制文本（copy text）。

然后不妨先退出 STATA，再重新打开 STATA。在命令窗口粘贴先前复制的命令，或者直接键入

```
. use "D:\Stata9\auto.dta", clear
```

在命令窗口中，回车即表示执行刚键入的命令，因此命令窗口不能回车换行。

```
. edit
```

这两行命令将再一次打开并显示 auto 数据集，同样点击右上角的 X 号，退出数据窗口。注意到，若不退出数据窗口，则命令窗口被禁止输入。

1.3.3 批量程序执行方式

注意到屏幕左上有一个 Review 窗口，该窗口记录使用过的命令，刚才键入的两个命令即出现在该窗口中，击活命令回顾窗口，点右键选择 save review content，在弹出的对话框中取名为 mydo 并保存，即得到程序操作文件。

先退出 STATA，然后重新打开 STATA，点击倒数第五个命令按钮，打开刚才保存过的程序文件 mydo.do，再点选 ，执行。

1.3.4 三种执行方式的相互关系

三种操作方式可以完成同样的任务。在初学命令的时候，当不记得某个命令时可以采用菜单操作方式得到该命令的用法。在结果窗口和命令回顾窗口都出现该命令，在命令窗口重复输入相应的命令，即可获得和窗口式操作同样的结果。如果将回顾窗口的命令保存，即得到程序，执行程序也得到同样的结果。

使用 STATA 时建议大家采用第三种方式，即写程序的方式，程序可以使得数据的处理和分析过程被完整保留下来，便于自己和他人进行修改和评论。

1.4 寻求帮助与网络资源

有多种途径可以获得 STATA 的帮助，主要的途径有三个：手册、STATA 自带帮助和网络帮助。对于多数人而言手册是可望不可及的，因为一套完整的手册有 10 余本，而且价格昂贵。但有了 STATA 的自带帮助，我们可以在记住极少

量的基本命令的基础上，方便地运用STATA命令¹。

1.4.1 获取帮助的命令

. help

显示出 STATA 所有帮助内容的目录结构。

如果输入具体的命令，则只显示该命令的帮助，如

. help summarize

也可以通过菜单式的点选方式获得帮助: Help>>stata command...在弹出的对话框中输入: summarize 然后回车，得到与 help summarize 同样的结果。

使用帮助的小窍门: 先看命令描述 (Description) 部分，然后直接看帮助文件后面的命令示例 (Examples)，将命令示例复制到命令窗口，执行，看看执行结果，体会命令的用法。

网络帮助可以采用如下命令获得

. findit scat3, net

. search scat3, net

这两条命令等价，均为寻找绘三维立体图的命令 scat3。由于 scat3 不是 STATA 内置命令，所以需要通过这两个命令搜索并下载安装后才能使用。

1.4.2 几个主要的网站

(1) STATA公司官方网站 <http://www.stata.com>

(2) STATA 资源链接 <http://www.stata.com/links/resources.html>

(3) STATA出版社 <http://www.stata-press.com>

(4) STATA电子杂志 <http://www.stata-journal.com/>

(5) STATA 技术公告版

<http://www.stata.com/support/faqs/>

<http://fmwww.bc.edu/gstat/docs/gsaFAQ.html>

<http://www.ats.ucla.edu/stat/stata/examples/default.htm>

1.5 命令示例

1.5.1 进行四则运算

. di 5+9

. di 5-9

. di 5*9

. di 10/2

. di 10^2

. di exp(0)

. di ln(1)

. di sqrt(4)

上述运算分别为加、减、乘、除、幂、指、对和开方，其中 di 为 display 的

¹ 最常用的命令见附录 2

简写，是一个 STATA 命令，该命令显示计算结果。

1.5.2 描述统计：求五数概略

任务：求价格和重量的观察值个数、平均值、标准差、最小值和最大值

```
. use auto, clear
```

```
. sum price
```

```
. sum weight
```

这两步命令也可以一步完成

```
. sum price weight
```

1.5.3 绘图

任务：绘出价格和重量的散点图和折线图

```
. scatter price weight
```

```
. line price weight, sort
```

1.5.4 生成新的数据

任务：生成新的数据 x , ($x=1, 2, \dots, 1000$); $y=x+100$.

```
. clear
```

```
. set obs 1000
```

```
. gen  $x=_n$ 
```

```
. gen  $y=x+100$ 
```

1.5.5 控制结果输出显示

```
. list  $n$ 
```

显示完一屏后会停住，此时按回车键和”l”会显示下一行；按”q”会终止命令，或者使用 ctrl+break；按其他键会显示下一页。

1.6 几个环境设置

1.6.1 设置屏幕滚动

在列示 1 到 1000 之前，若先设置 set more off，则屏幕不停止；反之 set more on 会使显示停止。

```
. set more off
```

```
. list
```

```
. set more on
```

```
. list
```

```
. q
```

1.6.2 清除内存中原有内容

```
. clear
```

1.6.3 设置内存大小

查看内存使用情况

. memory

设置内存

. set memory 10m

1.6.4 设置文件存取路径

在打开数据之前，先要定位数据的位置，其命令为

. cd d:/stata9

如果想知道当前路径下有哪些文件，可以用 **dir** 命令来列示

. dir

假设你想在 D 盘的根目录下创建一个新的文件夹 **mydata** 来存放数据文件，命令为 **mkdir**。

mkdir d:/mydata

然后，进入该目录，命令为 **cd**

. cd d:/mydata

1.6.5 错误提示

学会从 STATA 的错误提示中明白错在哪里非常重要。

. list myvar

上述命令试图显示变量 **myvar**，但是结果窗口仅出现如下的显示

variable myvar not found
r(111);

红色信息表明，没有找到一个叫 **myvar** 的变量，的确，我们的数据中并没有这个变量。List 巧妇难为无米之炊。

红色信息下面还有一个天兰色的 **r(111)**，用鼠标点击，即可弹进一个帮助信息框，给出错误的更详尽解释。

再比如，我们在求五数概略时，误把 **sum** 写成了 **sun**

. sun

unrecognized command: sun
r(199);

显示说不认识 **sun** 这个命令。

1.7 复习和练习

- (1) 复习本节学习的主要命令 **clear, help, list, whelp, use, search, cd**
- (2) 找到附录 2 中主要命令的帮助信息
- (3) 计算出 $8+2*\ln(100)-e^3/5^2$

1.8 附录

1.8.1 附录 1：本章命令的程序文件

```

=====chp1. do=====
clear                //清除内存
set memory 10m       //设置内存大小
cd d:/stata8         //在打开数据之前，先要定位数据的位置

use auto             //打开数据文件 auto.dta
*计算汽车的平均价格
sum price            //该步计算汽车的平均价格
sum price weight     //求价格和重量的观察值个数、平均值、标准差、最小
值和最大值

use /*使用系统中的数据*/ auto, clear
sum weight          ///
price               ///
length             //求重量、价格、长度的平均值
scatter price weight //绘价格和重量的散点图

clear                //清除内存
set obs 1000         //将数据指针定位到 1000 处
gen x=_n             //生成一个 1 到 1000 的边疆自然数，变量名为 x
gen y=x+100          //生成一个新的变量 y, y 从 101 到 1100

/*从上面的例子可见：
(1) 在最前面加上 “*” 号表示该行为注释语句，STATA 将只显示不执行；
(2) 在一个命令的中间加入注释，要用： /*注释内容*/
(3) 对较长的命令或者为便于阅读，将一行命令写成几行时，用///来分开
(4) 在命令行的后面加入注释： //注释语句
将一部分内容变成注释内容，前后用/*被注释掉的语句*/ */
=====end=====

```

1.8.2 附录 2: 常用命令

需求帮助

- **help**
- **search**

帮助

网络寻求帮助

进入某路径

- **cd**

设定内存

- **set memory 20m**

设置 STATA 的内存空间为 20m

打开和保存数据

- **clear**
- **use**
- **save**

清空内存数据

打开 STATA 格式的数据文件

保存内存中的数据

导入数据

- **input**
- **edit**
- **infile**
- **insheet**

录入数据

编辑数据

导入数据

导入数据

重整数据

- **append**
- **merge**
- **xpose**
- **reshape**
- **generate**
- **egen**
- **rename**
- **drop**
- **keep**
- **sort**
- **encode**
- **decode**
- **order**
- **by**

将有相同结果的数据纵向拼接（观察值拼接）

将两个数据文件横向拼接

数据转置

生成新的数据

生成新的数据

变量重命名

删除变量或观察值

保留变量或观察值

对观察值按从小到大顺序重新排列

数值型数据转换为字符型数据

字符型数据转换为数值型数据

变量顺序的重新排列

分类操作

报告数据

- **describe**
- **codebook**
- **list**
- **count**
- **inspect**
- **table**
- **tabulate**

总体展示数据情况

展示数据库中的每个变量情况

列示内存中的数据

报告共有多少观察值

报告变量的分布

数据列表

联列表

显示和保存输出结果

- **display**
- **log**

显示计算结果

将输出结果存放入结果文件

2 命令语句

2.1 掌握命令语句的格式

[by varlist:] command [varlist] [=exp] [if exp] [in range] [weight] [, options]

注：[]表示可有可无的项，显然只有 **command** 是必不可少的，下面结合例子分项来讲解命令的各个组成部分。

2.2 命令 **command**

[by varlist:] command [varlist] [=exp] [if exp] [in range] [weight] [, options]

```
. cd d:/stata9
. use auto, clear //打开美国汽车数据文件 auto.dta,后面的 clear 表示先清除内存中可能存在的数据集
. summarize /*很多命令可单独使用，单独使用时，一般是对所有变量进行操作，等价于后面加上代表所有变量的_all。 */
. summarize _all //注意到该命令输出结果与上一个命令完全一样
. sum //与前一命令等价，sum 为 summarize 的略写
. su //su 是 summarize 的最简化略写，不能再简化为 s
. s //简写前提是不引起混淆。执行这个命令将出现错误信息
```

unrecognized command: s

变量的省略规则

只要不引起歧义，命令可以尽量只写前几个字母。如 **summarize** 只需要前两个字母 **su**；而 **list** 只需要写第一个字母 **l**。在帮助文件中，命令下面有小划线，该线表明了命令可以省略到什么程度。如

list [varlist] [if] [in] [, options]

summarize [varlist] [if] [in] [weight] [, options]

练习：请用 **list** 进行仿照练习。

注意，在用 **list** 做练习的时候可能会遇到结果窗口停止，其右下角出现一个蓝色的“more”，按键盘上任何一个键，屏幕滚动一行。这一现象与第一讲中“set more on”的设置有关，请参考 1.6.1

2.3 变量 **varlist**

[by varlist:] command [varlist] [=exp] [if exp] [in range] [weight] [, options]

varlist 表示一个变量，或者多个变量，多个变量之间用空格隔开。

```
. cd d:/stata9
. use auto, clear
. sum price //求价格的观察值个数，平均值，方差，最小值和最大值
. su p //变量和命令均可略写,注意到两个结果完全一样
. su t //分数据中有两个变量的开首字母为 t (trunk 和 turn)，所以 STATA 认为 t 为模糊的省略。
```

m ambiguous abbreviation /红色为错误信息

```
. sum tr tu //求 trunk 和 turn 变量的五数概略统计
```

`. su t*` //等价于前一命令，以 t 开首的所有变量可用 t*来表示。

变量名称

除以下字符不能用作变量名外，任何字母、字母与数字(单独的数字也不允许)组合均可用做变量名：

`_all _b byte _coef _cons double float if in int long _n _N _pi _pred _rc _se _skip using with`

基本要求如下：

_ 第一个字元可以是英文字母或，但不能是数字；

_ 最多只能包括32 个英文字母、数字或下划线；

_ 由于 STATA 保留了很多以 “_” 开头的内部变量，所以最好不要用为第一个字元来定义变量。

2.4 分类操作 by varlist

[by varlist:] command [varlist] [=exp] [if exp] [in range] [weight] [, options]

先看下面的例子, 在汽车数据集中, 有一个变量 foreign, 该变量表示某个车是进口车 (1 Foreign) 还是国产车 (0 Domestic)。如果我们需要知道车的平均价格和重量, 则

```
. cd d:/stata9
```

```
. use auto, clear
```

```
. sum price weight
```

*如果需要分别知道国产车和进口车的价格和重量, 可以采用分类操作来求得,

```
. by foreign: sum price weight //分别计算国产车和进口车的价格和重量
```

但如果执行下面两个命令, 将出现错误*/

```
. sort price //按价格从低到高重新排序
```

```
. by foreign: sum price weight
```

***not sorted**

/* 系统提示没有排序, 这是因为 by varlist 在执行时要求内存中的数据是按照 by 后面的变量排序的。当我们用 sort price 重新排序后, 就打乱了原来按照 foreign 的排序, 所以出现了错误提示。更正的办法是: */

```
. sort foreign //按国产车和进口车排序
```

```
. by foreign: sum price weight
```

*更简略的方式是把两个命令用一个组合命令来写。

```
. by foreign, sort: sum price weight
```

如果不想从小到大排序, 而是从大到小排序, 其命令为 gsort。

```
. sort - price //按价格从高到低排序
```

```
. sort foreign -price /*先把国产车都排在前, 进口车排在后面, 然后在国产车内再按价格从大到小排序, 在进口车内部, 也按从大到小排序*/
```

2.5 赋值及运算=exp

[by varlist:] command [varlist] [=exp] [if exp] [in range] [weight] [, options]

该选项主要用于给新变量赋值或替换原变量的值

例: 生成一个新的价格变量 nprice, 该变量的取值为原汽车价格变量 price 的

基础上涨 10 元

```
. cd d:/stata9
. use auto, clear
. gen nprice=price+10    //生成新变量 nprice, 其值为 price+10
. list price nprice      //比较一下两个变量的取值
```

/*上面的命令 generate(略写为 gen) 生成一个新的变量, 新变量的变量名为 nprice, 新的价格在原价格的基础上均增加了 10 元。

```
. replace nprice=nprice-10    /*命令 replace 则直接改变原变量的赋值,
                                nprice 调减后与 price 变量取值相等*/
. list price nprice          //再比较一下两个变量, 相等。
```

2.6 条件表达式 if exp

[by varlist:] command [varlist] [=exp] [if exp] [in range] [weight] [, options]

例：若只想查看国产车的品牌和价格，则加入筛选条件 `if foreign==0` */

```
. cd d:/stata9
. use auto, clear
. list make price if foreign==0
```

*只查看价格超过 1 万元的进口车（同时满足两个条件），则

```
. list make price if foreign==1 & price>10000
```

*查看价格超过 1 万元或者进口车（两个条件任满足一个）

```
. list make price if foreign==1 | price>10000
```

*分类型查看价格超过 1 万元的汽车的品牌和价格

```
. by foreign, sort: list make price if price>10000
```

2.7 范围筛选 in range

[by varlist:] command [varlist] [=exp] [if exp] [in range] [weight] [, options]

如果要计算较低的前 10 台车的平均价格，则要先按价格排序，然后仅对前 10 个车的价格求平均值

```
. cd d:/stata9
. use auto, clear
. sort price
. sum price in 1/5
```

注意“1/5”中，斜杠不是除号，而是从 1 到 5 的意思，即 1, 2, 3, 4, 5。

如果要计算前 10 台车中的国产车的平均价格，则可将范围和条件筛选联合使用。

```
. sum price in 1/10 if foreign==0
```

2.8 加权 weight

[by varlist:] command [varlist] [=exp] [if exp] [in range] [weight] [, options]

任务：下表是 2005 年湖北省高考 640 分及以上成绩一分一段的人数统计，第一列 score 为高考分数，第二列 num 为该分数段的人数。现在我们要求 640 分及以上考生的平均分数。

score	num
650	193
649	26
648	23
647	16
646	21
645	26
644	32
643	23
642	38
641	29
640	38

操作：

先将上面的表格复制，然后进入 STATA，执行如下命令

```
. clear           //清空 STATA
```

```
. edit
```

然后把光标定位在表格的第一行第一列，点右键，选择粘贴 (paste)，上表数据便被复制到 STATA 中，退出数据编辑器。

```
. sum score       //思考：得到的结果是 640 分及以上考生的平均分吗？
```

简单地使用 sum 命令得到的平均成绩显然是不正确的，因为各个分数下的人数是不一样的，正确的计算需要加权，加权的办法是

```
. sum score [weight=num] /*加权计算，比较该结果与 sum score 的区别，
                           实际上，不用权重选项时，相当于权重相等。*/
```

```
. sum score [w=n]       //w 为 weight 的略写，n 为 num 的简写，两命令等价
```

2.9 其他可选项,options

[by varlist:] command [varlist] [=exp] [if exp] [in range] [weight] [, options]

许多命令都有一些可选项

例如，我们不仅要计算平均成绩，还想知道成绩的中值，方差，偏度和峰度等*/

```
. sum score, detail
```

```
. sum score, d       //d 为 detail 的略写，两个命令完全等价
```

注意，结果中显示了 1%，5%等分位数，意思是把变量从小到大排序，第 1%位置处的取值是多少，第 10%的位置上的取值是多少。显然，50%位置处的取值是中位数。此外，加了 detail 选项后，还得到最小的前 5 个数，最大的 5 个数，以及峰度和偏度等。

*再如，list 命令也有一些可选项

```
. cd d:/stata9
```

```
. use auto, clear
```

```
. list price in 1/30, sep(10)    //每 10 个观察值之间加一横线
```

```
. list price in 10/30, sep(2)    //每 2 个观察值之间加一横线
```

```
. list price, nohead            //不要表头
```

2.10 复习与练习

对照本章的命令结构，用帮助命令打开第一讲中附录 2 中常用命令的相应帮助文件，对照帮助文件下面的例子体会各种命令的用法。

3 数据

数据文件是一个矩形的矩阵，这个矩阵的每一行都代表或对应着一个“观测单位”（如张三、李四、王五），矩阵的每一列都代表或对应着一个“变量”（比如年龄，身高、体重，月工资收入等等）。因此，数据文件矩阵中的每一个元素（case）都代表或对应着某一个“观测单位”中的某一个“变量”的变量值或观察值。

3.1 打开示例数据和网络数据：use

3.1.1 示例数据

示例数据为 STATA 帮助文件中所用的数据，其后缀名为.dta，如果在 STATA 软件当前路径下，直接用 use 命令即可打开；如果不在当前路径下，则可以使用 sysuse 命令打开。

```
. use auto,clear           //打开汽车数据 auto.dta
. cd d:/                   //改变路径到 d:/
. use auto, clear
file auto.dta not found  //系统提示无法找到文件, 因为 auto.dta 不在 d:/
r(601);
. sysuse auto,clear       //无论当前路径是什么, 该命令均能打开系统自带文件
```

示例数据

Auto :美国 1978 年汽车数据，包括产地、车名、行使里程、重量等变量
 Bplong 血压数据
 Cancer 药物实验生存数据
 Census1980 年美国分州人口普查数据
 Citytemp 美国城市气温数据
 Educ99gdp 教育与 GDP 关系数据
 Gnp96 美国 1967-2002 年的 GNP 数据
 Lifeexp 预期寿命数据
 Nlsw88 美国年轻妇女研究数据
 Pop2000 美国 2000 年人口普查数据
 Sp500 S&P500 历史数据
 Uslifeexp 美国预期寿命 1900-1999
 Voter 美国 1992 年选举民意调查数据



3.1.2 从网络获取数据

上述示例数据可能没有全部下载到你的所用的电脑中，因此简单地使用 use 和 sysuse 命令时，可能出现错误，如

```
. use nlswork, clear
file nlswork.dta not found
```

此时，如果确定该数据为示例数据，可以直接通过网络获取，其命令为：

`. use http://www.stata-press.com/data/r9/nlswork` //从网站获取数据，或者
`. webuse nlswork, clear` //与前一命令等价，从 STATA 官方数据库获取数据
 webuse 只能从 `http://www.stata-press.com/data` 这一路径获取数据，如果不是该网站的数据，webuse 失效，只能把网站地址完全写出来。使用该命令时必须确保网络连接正常。

另一个网络数据较多的地方是波士登大学的数据中心，伍德里奇的《计量经济学导论》一书中所使用的全部数据都可以通过该数据中心获得。比如

`. use http://fmwww.bc.edu/ec-p/data/wooldridge/CEOSALI`

即打开教材中例 2.3 中所使用的 CEO 数据。

use 命令只能打开后缀名为“*.dta”格式的数据，.dta 格式以外的数据，STATA 不能直接读取，需要从外部读入，最简单而直接的办法是复制和粘贴。但是有时没有其他软件，比如，我们有 SAS 格式或 SPSS 格式的数据，但没有 SAS 软件和 SPSS 软件，此时需要用 STATA 提供的其他命令或者使用 transfer 数据格式转化软件。在讨论其他输入或导入数据的方法之前，我们先来学习一点数据类型的知识。

3.2 数据类型

STATA 通常把变量划分为三类：分别是数值型，字符型和日期型

3.2.1 数值变量：

用 0、1、2...9 及+、-（正负号）与小数点“(.)”来表示。在输入数据时，逗号不能被识别，如 1,024 应该直接写成 1024. 其他示例

5
 -5
 5.2
 5.2e+3
 5.2e-2

后面两个数据为科学计数法的数据，分别表示 5200 和 0.052. 其中的 e 相当于 10，因此 5.2e+3 的意思是： $5.2 \times 10^3 = 5200$

数值型变量按其精度区分，又有五种类型，分别是：

存贮类型	最小	最大	0-领域	字节
byte	-127	100	+/-1	1
int	-32,767	32,740	+/-1	2
long	-2,147,483,647	2,147,483,620	+/-1	4
float	$-1.70141173319 \times 10^{38}$	$1.70141173319 \times 10^{36}$	+/- 10^{-36}	4

```
double -8.9884656743*10^307 8.9884656743*10^307 +/-10^-323 8
```

当运算精度要求很高的时候，需要将变量设置成浮点型或双精度型。

另注意 1 和 1.0000 的精度是不同的，前者在 (0.5, 1.5) 区间内近似，而后者在 (0.99995, 1.00005) 区间内近似。若多次运算反复取四舍五入，精度较低时将使计算误差迅速变大，然而，精度高时占用的内存资源较多。下面的命令有助于理解变量存储类型变换。

```
. clear
. set obs 1 //将设定一个观察值
obs was 0, now 1 //提示信息说,之前系统中没有观察单位,现在有了一个
. gen a=1 //生成一个新变量 a,令 a 取值为 1
. d /*d 为 describ 命令的略写, describ 命令显示数据集的
    属性信息,注意观察显示结果中, a 的 storage type 为 float 型,
    浮点型为默认类型*/
```

Contains data

```
obs:      1
vars:      1
size:      8 (99.9% of memory free)

      storage   display   value
variable name  type     format   label       variable label
a              float    %9.0g
```

Sorted by:

Note: dataset has changed since last saved

```
. compress //在不损害信息的基础上压缩,使数据占用空间尽可能小
a was float, now byte //a 由浮点型变为了字节型
. d //注意 a 的 storage type 现在为 byte 型
. replace a=101 /*注意 a 的 storage type 现在自动升为 int 型,
                因为 byte 最大只能为 100*/
a was byte now int
(1 real change made)
. replace a=100
. compress
. d //重新变回到 byte 型
. replace a=32741 //直接变到 long 型,因为 int 型最大只能到 32740
. gen double b=1 //直接生成双精度变量 b
. recast double a //将 a 变成双精度变量 b
. d //注意到 a 和 b 均为双精度型
```

3.2.2 字符串变量

字符变量通常是一些身份信息，如姓名，地名。另外，定类变量也可以用字符变量来表示，如性别分为“男”和“女”。

字符串变量由字母或一些特殊的符号组成（如地名〈籍贯〉变量，迁出地，住址，职业等等）。字符串变量也可以由数字来组成，但数字在这里仅代表一些符号而不再是数字。字符串变量通常以引号“”注标，而且引号一般不被视同为字符的一部分，注意这里的引号必须是英文输入状态下的引号。

字符串最多可以达 244 个字符。一般用 str#来表示字符的多少，如 str20 表示将有 20 个字符。一般三个中文字的姓名需要 6 个字符。

字符型示例

```

“String”
“string”
” string”
”string ”
”” //特殊字符串，表示空字符，缺失值。
” ” //注意与空字符串的区别，含有一个空格
”125.27” //” 125.27” 由于有双引号，将被视同为字符而非数值。
“$2,343.68”
“I love you”
“旺材是条狗”

```

注意前四个字符串均不相同，大小写是不一样的，有无空格及空格的位置不同，都表示不同的字符串。对于” 125.27” 这样的数值型的字符串，可以用 real() 函数或者 destring 命令转化成数值型变量。具体操作见 3.3.1。

3.2.3 日期型变量

在 STATA 中，1960 年 1 月 1 日被认为是第 0 天，因此 1959 年 12 月 31 日为第-1 天，2001 年 1 月 25 日为 15000 天。对日期型变量的讨论将在后面的时间序列分析部分，见第 16 讲。

```

1999 12 10
jan/10/2001
10jan2001
...
-15,000 --- 01dec1918
-31 ----01dec1959
...
-1 --- 31dec1959
0 --- 01jan1960
1 ---- 02jan1960
...

```

```

31 ---- 01feb1960
...
15,000 ---- 25jan2001

```

3.2.4 缺失值

没有意义的计算结果显示为“.”

. display 2/0

另一种情况是，数据中含有缺失值，而 STATA 默认的缺失值也用“.”来表示。在有些数据文件中，缺失值不是用“.”或者空来表示的，而是用-9996 等来表示，如果要将其全部替换为“.”，或者反之，将“.”替换为-9996，命令为：

```

. mvencode age,mv(-9996)
. mvdecode age,mv(-9996)

```

3.3 数据类型转化

任务: 将 destring1, destring2 和 tostring 中的数据类型进行相互转化

*3.3.1 字符型转化成数值型: destring

*destring1 数据中的数据全为字符型，转换为数值型

.webuse destring1, clear

.des /*注意到所有的变量存储类型 (storage type) 均为字符型 str#, 其中#号表示字符串长度*/

Contains data from <http://www.stata-press.com/data/r9/destring1.dta>

obs: 10

vars: 5 3 Mar 2005 10:15

size: 240 (99.9% of memory free)

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

id	str3	%9s		
----	------	-----	--	--

num	str3	%9s		
-----	------	-----	--	--

code	str4	%9s		
------	------	-----	--	--

total	str5	%9s		
-------	------	-----	--	--

income	str5	%9s		
--------	------	-----	--	--

.sum //因为所有变量为字符型，所以不能进行数值计算

.gen nincom=incom+10 //因字符不能进行四则运算，不能进行加法运算

***type mismatch** //系统提示类型不匹配，因为 income 为字符型，10 为数值型

.destring, replace //全部转换为数值型，replace 表示将原来的变量（值）更新

.sum //注意到转换为数值型后，可以求五数概略了

.gen nincom=income*1.3 //转换后，可以运算，工资终于涨了 30%!


```
.list nincom income    //工资终于涨了 30%!
```

-----将字符型数据转换为数值型数据: 去掉字符间的空格-----

*destring2 数据集中的 data 变量为字符型, 且年月日间有空格, 转移为数据型

```
.webuse destring2, clear
.des                      //注意到所有的变量均为字符型 str
.list date                //注意到 date 年月日之间均有空格
date
-----
1.  1999 12 10
2.  2000 07 08
3.  1997 03 02
4.  1999 09 00
.destring date, replace   //想把 date 转换成数值型, 但失败了, 系统提示说
*date contains non-numeric characters; no replace /*由于含有非数值型字符
(即空格), 因此没有更新, 也即转换命令没有执行。*/
.destring date, replace ignore(" ") /*忽略空格, 然后转换, 注意这里的" "中
间有一个空格, 不是""。*/
date: characters space removed; replaced as long //成功转换为 long 型
.des                      //注意到 date 的 storage type 已变为 long
.list date                //注意到空格消失了
date
-----
1.  19991210
2.  20000708
3.  19970302
/*与 date 变量类似, 变量 price 前面有美元符号, 变量 percent 后有百分号,
换为数值型时需要忽略这些非数值型字符。*/
.destring price percent, gen(price2 percent2) ignore("$,%")
.list                      //注意到 price2 前面的$号消失, percent2 后面的%号消失
      date      price      price2      percent      percent2
-----
1.  19991210      $2,343.68      2343.68      34%      34
2.  20000708      $7,233.44      7233.44      86%      86
.d                      //注意到 price2 和 percent2 均变为数据值型变量 double 和 byte
```

*3.3.2 数值型转化为字符型: tostring

```
.webuse tostring, clear /*该数据中年月日的数据类型不一样, 不能直接相加
生成一个反映日期的新变量*/
.des                      //注意到 month 为字符型, 而年和日为数值型
.list
.gen date1=month+"-"+day+"-"+year //将年月日构成一个新的日期变量
type mismatch //由于 month 为字符型, 年和日为数值型, 不同类型不能相加
r(109);
```

```
.tostring year day, replace //将年和日转化为字符型
.des //注意到,现在全部变为字符型
.gen date1=month+"/"+day+"/"+year //将年月日构成一个新的日期变量
.list //生成了一个新的变量 date1, 其为三个字符串和两个"/"符号连接而成
.gen date2=date(date1,"mdy") /* date() 为日期函数, 它以 1960 年 1 月 1
    日为第 0 天, 计算从那天起直到括号中指定的某天 date1
    一共过了多少天。"mdy"指定 date1 的排列顺序, 这里是
    按照月日年的顺序来表示日期。*/
.list //新生成的 date2 表示总天数
*小游戏: 请算算你活了多少天? 示例: 一个生于 1975 年 12 月 27 日的家伙,
他活了?
.di date("1975/12/27","ymd")
```

3.4 数据显示格式:format

/*format 只控制数据的显示格式, 并不改变内存中数据的大小。*/

```
.webuse census10,clear //美国人口普查数据
.des //第三列显示了数据的格式 display format
```

variable name	storage type	display format	value label	variable label
state	str14	%14s		State
region	int	%8.0g	cenreg	Census region
pop	long	%11.0g		Population
medage	float	%9.0g		Median age

*注意到, stata 变量的格式为 %14s, 表示右对齐, 共 14 个字符, % 为固定用法

```
.list in 1/4 //注意不同的显示格式:均为右对齐
      state      region      pop      medage
```

```
-----
1.  Alabama      South      3893888      29.3
2.   Alaska      West       401851      26.1
3.   Arizona      West      2718215      29.2
4.  Arkansas      South     2286435      30.6
```

```
.format state %-14s // 该命令使 stata 的显示格式左对齐, 14 前面多了个负号
```

```
.list in 1/4 //注意不同的显示格式, state 现在左对齐了
```

```
+-----+
      state      region      pop      medage
-----+-----
1.  Alabama      South      3893888      29.3
2.   Alaska      West       401851      26.1
3.   Arizona      West      2718215      29.2
4.  Arkansas      South     2286435      30.6
```

```
.format region %-8.0g /*region 变量看起来是字符型变量, 但实际上为
                        数据型, 它也可以左对齐, 同样是加一个负号.
```

`.list in 1/4` //注意 region 现在左对齐了

`.format pop %11.0gc` /*pop 的显示格式为%11.0g,后面加上 c,则每三位数间用逗号分开,c 为 comma 的意思.*/

`.list in 1/4` //结果加上了逗号,但是第五个观察值没有任何变化

	state	region	pop	medage
1.	Alabama	South	3,893,888	29.3
2.	Alaska	West	401,851	26.1
3.	Arizona	West	2,718,215	29.2
4.	Arkansas	South	2,286,435	30.6
5.	California	West	23667902	29.9

*因为这个数太大,加逗号将超过 11 位数,我们可以先把总的位数增加

`.format pop %12.0gc` //把 pop 显示总长度数增加到 12 位

`.list in 5` //现在所有的 pop 都按逗号分开了

5.	California	West	23,667,902	29.9
----	------------	------	------------	------

`.format medage %8.1f` //要求所有的 medage 都显示一位小数

`.list in 1/4`

	state	region	pop	medage
1.	Alabama	South	3,893,888	29.3
2.	Alaska	West	401,851	26.1
3.	Arizona	West	2,718,215	29.2
4.	Arkansas	South	2,286,435	30.6

`.gen id=_n` //生成一个新变量 id,取值依次为 1, 2, 3

`.replace id=9842 in 3` //将 id 的第三个变量替换为 9842

`.list in 1/3`

	state	region	pop	medage	id
1.	Alabama	South	3893888	29.3	1
2.	Alaska	West	401851	26.1	2
3.	Arizona	West	2718215	29.2	9842

`.format id %05.0f` //对于编号,我们希望前面用零使得位数对齐

`.list in 1/3` //注意到通过在前面补零,所有的 id 都成了 5 位数。


	state	region	pop	medage	id
1.	Alabama	South	3893888	29.3	00001
2.	Alaska	West	401851	26.1	00002
3.	Arizona	West	2718215	29.2	09842

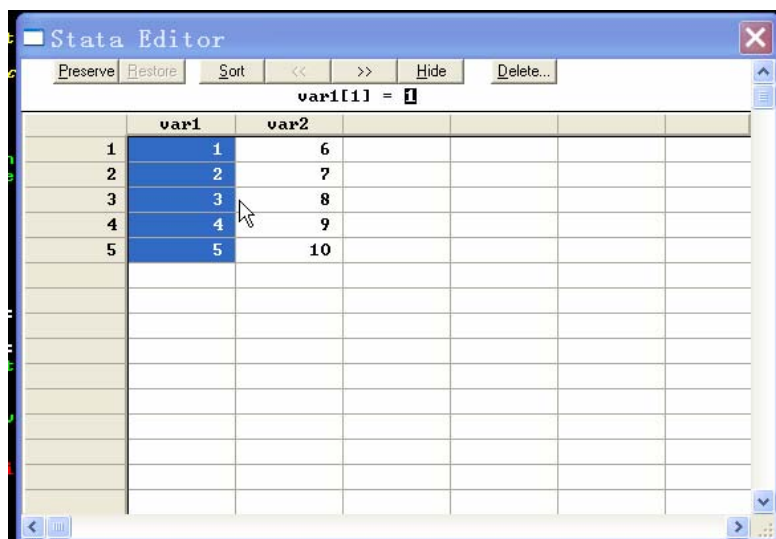
3.5 在 STATA 中直接录入数据: input

3.5.1 菜单式操作

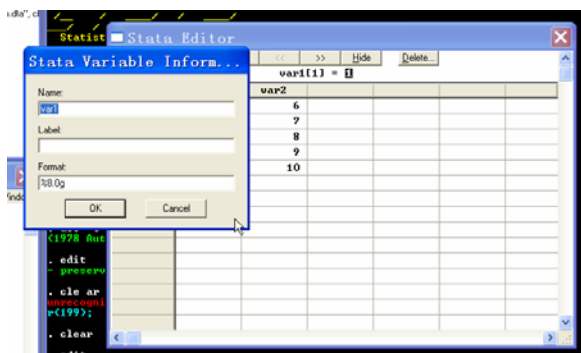
任务: 按学号录入五个学生的经济学成绩


id	economy
1	40
2	80
3	90
4	70
5	53

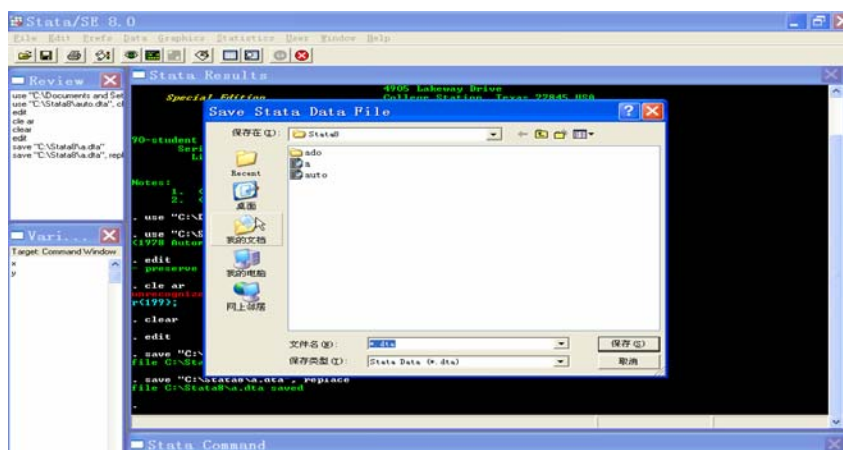
操作：（1）点击  图标>>在打开的数据表格第一列中录入五个姓名>>在第二列中录入另五个成绩



双击 var1 弹出对话框>>将变量改名为 id>>在 label 中写入学号>>退出弹出窗口；
双击 var2 弹出对话框>>将变量改名为 name>>在 label 中写入姓名>>关闭数据编辑器



（2）点击  图标保存数据>>给数据命令为 student>>退出



在建立数据文件后，如果没有存盘，这个文件即是一个“临时的”数据文件，它将随着退出 **STATA** 系统时而消失。当数据文件被存储在后，它将成为一个“永久性”的数据文件，用户可以在以后经常使用它而不必重新建立之。

3.5.2 命令操作

任务：按学号录入五个学生的学号和姓名

id	name	economy
1	John	40
2	Chris	80
3	Jack	90
4	Huang	43
5	Tom	70

操作：在 `command` 窗口中键入(注：前面的点号不必键入，每完成一行按回车键，黑体为命令，斜体为变量名或文件名)：对于字符型变量，需要指明其为字符型并指明最大的字符长度。

- **clear** //清空内存
- **input id str10 name economy** //输入变量名，特别注意姓名前的 str10.
- 1 John 40 //录入第一个学生的学号和成绩
- 2 Chris 80 //录入第二个学生的学号和成绩
- 3 Jack 90
- 4 Huang 70
- 5 Tom 53
- **end** //录入数据结束
- **save economy** //保存数据到当前路径，文件名为 economy

3.5.3 程序操作

(1) 打开 do file editor, 键入以下内容:


```
clear           //清空内存
input id str10 name economy //输入变量名，特别注意姓名前的 str10.
1   John 40      //录入第一个学生的学号和成绩
```

```

2 Chris 80
3 Jack 90
4 Huang 70
5 Tom 53
end           //录入数据结束
save economy,replace //保存数据到当前路径，文件名为 economy

```

(2) 保存程序文件为 mydo

(3) 点击，执行后得到数学成绩

3.6 导入其他格式数据：insheet

经常会遇到的情形是：我们有其他格式的数据，需要导入到 STATA 中进行分析，建议大家此时将其他格式数据复制到分析数据的文件目录下，然后直接用 STATA 的导入数据文件命令导入原始数据，用程序模式进行处理，然后导出处理结果。这样做的最大好处是：既不会破坏最原始的数据文件，又使我们的每一步数据处理和分析过程都有迹可循。

3.6.1 insheet 命令

在本书所附数据文件中找到“3origin.xls”数据，将其打开并另存为“3origin.csv”，(另存时请注意要选择“保存类型”下拉单，选择 CSV（逗号分隔）这一项)。然后在 STATA 命令窗口中用下述命令导入

```
. insheet using 3origin.csv, clear
```

也可以先将“3origin.xls”数据打开并另存为“3origin.txt”，然后用下面的命令导入

```
. insheet using 3origin.txt, clear
```

当数据中某个变量的位数特别长或者对导入数据的精度要求很高的时候，需要在该命令后面加 double 选项。

```
. insheet using 3origin.txt, double clear
```

3.6.2 infile 命令

对于“3origin.txt”或“3origin.csv”，还可用 infile 命令导入 STATA，此时需要先指出变量名。尤其要注意，当变量为字符型时，要先指明。

```
infile id str10 name gender minority economy math using origin.txt, clear
```

或者

```
infile id str10 name gender minority economy math using origin.csv, clear
```

3.6.3 infix 命令

还有一种标准化的数据，每个变量的位数是确定的，不足时，前面用 0 补齐，以 origin.数据的后面四个变量为例，其数据格式为

```

114068
128052
029076
024390

```

037096

115385

028536

129565

如果遇到这种数据格式，需要对照数据说明导入数据，相应的命令为：

infix gender 1 minority 2 economy 3-4 math 5-6 using origin.csv, clear

其中的数字为对应的数字位数。

3.6.4 outsheet 命令

与前述三个命令相反，有时我们需要将 STATA 数据导出为其他格式数据，比如文本格式或后缀为 acs 的格式：此时需要使用 outsheet 命令实现，该命令的基本格式如下。

outsheet using myresult.txt

outsheet using myresult.asc

此时建立的文件 myresult.txt 第一行为变量名，第 2~6 行为变量值。变量列间用 Tab 键分隔。如果不希望在第一行存储变量名，则可以使用 nonames 选项。如果文件已经存在，则需要使用 replace 选项，相应的命令分别为。

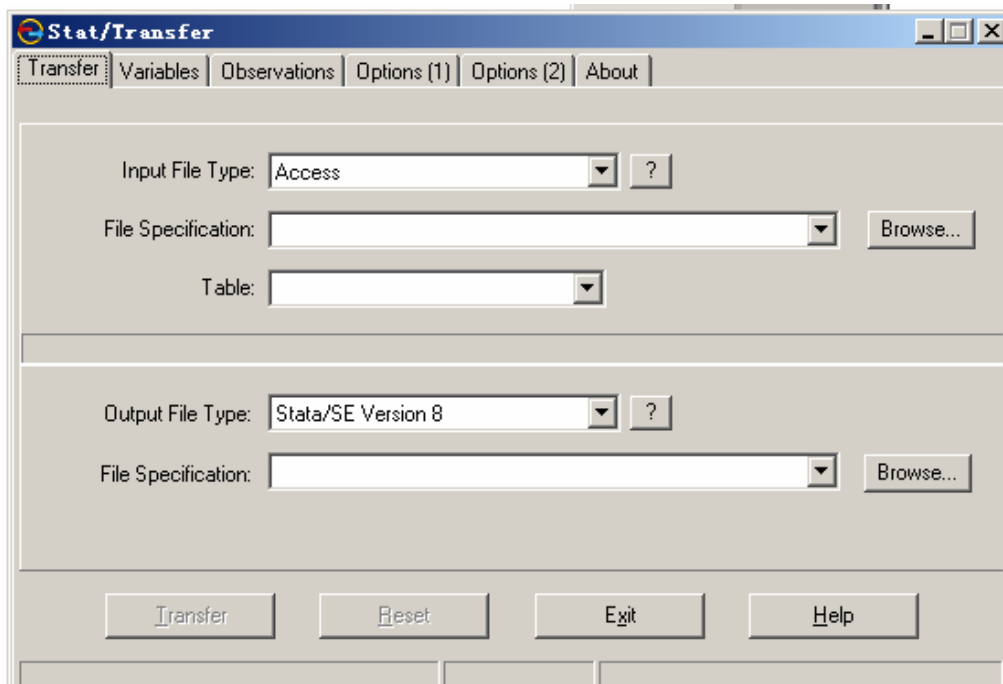
outsheet using myresult.asc, nonames

outsheet using myresult.asc, nonames replace

3.6.4 使用 transfer 软件

Transfer 软件专用于转换不同格式的数据文件，使用起来非常方便。只需要在 input File Type 栏中选择需要转化的原数据文件类型，然后定位打开需要转化的原数据文件。再选定输出文件类型，指定输出文件的存放位置和文件名。最后点击 transfer 按钮。数据便被转化。

该软件可在 <http://www.pinggu.org/bbs> 上下载试用，不过做正式工作，建议采用正版软件。



3.7 标签数据:label

要掌握的命令：为了创建一个完整的文件，要掌握下面的命令。

命令	命令解释	用法示例
pwd	显示当前路径	pwd
dir	列示当前路径文件夹中的所有文件	dir
mkdir	在当前路径下创建一个新的文件夹	mkdir d:/mydata
cd	将 cd 后面的路径设定为当前路径	cd d:/mydata
describ	显示整个数据集的信息	des
rename	将现有变量名改为新的变量名	rename gender sex
label	给数据/变量/变量值加注标签说明	
label data	标签数据	label data “2004 级成绩表”
label var	标签变量	label var name “姓名”
label value label define	标签变量值	label values gender genderlb label define genderlb 1 "男" 0 "女"
note	为数据加注额外说明	note: 9 月 10 日为数据加注说明
list	列示内存中的数据	list id name
save	保存数据	save mydata, replace
erase	删除数据文件	erase mydata1.dta, replace

以上命令可以通过 **help command** 查看到具体的命令格式。

要完成的任务：创建一个文件（文件名为 mydata.dta）并将其放在 D: /mydata 文件夹下（如果没有该文件夹，请用 mkdir 命令创建）。标签该数据（用 label 命令）使得任何一个使用该数据的人都能明白该数据（包括整个数据/其中的变量及变量值）的含义。

原始数据的内容如下：3origin.xls

1	John	1	1	40	68
2	Chris	1	2	80	52
3	Jack	0	2	90	76
4	Huang	0	2	43	90
5	Tom	0	3	70	96
6	Han	1	1	53	85
7	Phillip	0	2	85	36
8	Jin	1	2	95	65

其中第一列为学号 id，第二列为姓名 name，第三列为性别 gender（1 表示男性，0 表示女性），第四列为民族 minority（1 表示汉族，2 表示少数民族，3 表示不知道）；第五列为经济学成绩 economy，第六列为数学成绩 math。

案例的参考操作

注：红色双线之间的内容可以复制到 STATA 的程序编辑器中直接执行，执行到红色错误信息处会停止，如要继续执行，可选定后面的内容接着执行。

另外，下面程序中的所有命令中加黑的为固定用法，不能灵活选择；而斜体表示

文件名和变量名，可以自己根据自己的文件名和变量名情况灵活选择或改动。

***=====chapter2.do=====**

***———————路径和环境设定———————**

/*假设你想在 D 盘的根目录下创建一个新的文件夹 mydata 来存放数据文件，命令为 mkdir。如果该文件夹已经存在，运行该命令时会出现错误信息，加上 capture 后，STATA 会自动判断，如果 mydata 文件夹存在，则跳过该命令，如果不存在，则创建。*/

clear

capture mkdir d:/mydata

*然后，进入该目录，命令为 cd

cd d:/mydata

***3.7.1 变量重命名：rename**

/*采用直接复制粘贴法，将原始数据粘入 stata。*/

然后退出数据编辑器，先将该数据保存起来，文件名为 3origin.dta。*/

save 3origin, replace

/* 新粘入的变量自动命令为 var1, var2,...var6，为使变量容易理解和记忆，要将变量重新命名，命令为 name。在此之前，可以先用一个 describ 命令看看数据情况*/

use 3origin, clear

des //查看数据集的整体情况，注意变量名为 var1-var6

renprefix var v //将所有 var 开头的变量名改为以 v 开头

rename v1 id //将第一个变量重新命令为 id

rename v2 name

rename v3 gender

rename v4 minority

rename v5 economy

rename v6 math

des //再次查看数据集的整体情况，注意变量名已改变

***3.7.2 标签文件：label data**

/* 为避免时间太长，忘记变量的含义，我们可以用 label 命令来标记。该命令可以用来标记数据文件，如将文件取名为“2007 年秋 5632 班学习成绩单”*/

label data “2007 年秋 5632 班学习成绩单”

* 在文件处理过程中加注说明，命令为 notes

note: 2007 年 9 月 6 日由任我行创建该数据

* 下一次打开数据，要查看创建和数据处理的说明时，直接键入

note

***3.7.3 标签变量：label var**

*也可以用 label 命令来标记变量，如将 id 标记为“学号”

label var id “学号”

label var name “姓名”

```
label var gender “性别 1=男 2=女”
```

```
label var minority “民族”
```

*3.7.4 标签变量值：label define 和 label values

*还可以标记变量的取值。注意要按以下两步来操作：

```
label define genderlb 1 “男” 0 “女”
```

```
list //注意此时 gender 变量显示的值为 0 或 1
```

```
label values gender genderlb //该命令仅仅是显示的变化，实质不变
```

```
list //注意此时 gender 变量显示的值为男或女
```

```
label define minoritylb 1 “汉族” 2 “少数民族”
```

```
label values minority minoritylb
```

```
list
```

*3.7.5 标签增加与修改：add 和 modify

/* 定义完汉族和少数民族后发现还有些学生的民族是不知道的（原始值为 3），则*/

```
label define minoritylb 3 “不知道”
```

*然而结果窗口却显示出如下错误信息，

```
label minoritylb already defined
```

* 因为 minoritylb 已经存在并被定义，我们需要加上选项, add

```
label define minoritylb 3 “不知道”, add
```

```
list
```

```
label define minoritylb 3 “don't know”, add
```

/*试图将“不知道”改为英文“don't know”，再次显示错误

```
*invalid attempt to modify label
```

因为 3 已被定义，这次不是增加而是修改，所以选项为, modify*/

```
label define minoritylb 3 “don't know”, modify
```

```
list
```

*3.7.6 标签显示与删除：dir 和 drop

```
label dir //显示标签
```

```
label list //显示标签的赋值含义
```

```
list //注意到 minority 显示的是“汉族”、“少数民族”和 “don't know”
```

```
label drop minoritylb //删除标签
```

```
label dir //再显示标签
```

```
list //注意到 minority 显示的是 1, 2, 3
```

*3.7.7 保存和删除数据文件：save 和 erase

```
compress //压缩数据，使之在不损失任何信息的前提下占用空间最小
```

```
save mydata //保存数据，数据文件名为 mydata
```

*如果同一文件夹下已经存有 mydata.dta, 而你又要再次执行 save mydata 时，系统会出现提示

```
save mydata
```

file mydata.dta already exists

*该提示表示数据已经存在, 此时我们可以换名保存

save mydata1

*或者将原文件覆盖, 方法是加上 replace 选项

save mydata, replace

*删除文件, erase

erase mydata1.dta

*注意: 删除文件时一定要带上后缀名。

*=====end=====

3.8 复习与练习

(1) 建立个人信息数据库, 文件名为 st+学号

id 学号	Name 姓名	Gender 性别	age 年龄	Prov 家乡省

(2) 建立自己所在省数据库: 从网上收集家乡所在省各年的 GDP 数据和人口数据, 并将其转化为*.dta 数据文件, 名为 pv+学号

省调查表				
学号	家乡省份	省历年 GDP	省历年人口	省农村居民人均纯收入

(3) 复习: 本节学习的主要命令 input, end, save, exist, clear, list, help, use, search, use webuse sysuse insheet infile infix

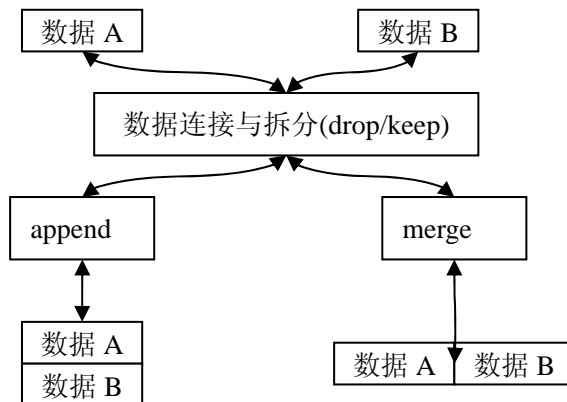
4 数据整理

4.1 拆分与连接数据文件要掌握的命令

为了拆分或合并数据文件，要掌握下面的命令。这也是该小节的学习目标。

命令	命令解释	用法示例
drop	删除变量或观察值	drop math
keep	保留变量或观察值	keep math
append	将两个数据集拼接(观察值拼接)	append using math
merge	将两个数据集合并(变量合并)	merge id using math
reshape	将数据重整	reshape long inc, i(id) j(yr)
stack	将多列数据转换成一列数据	stack a b c d, into(e f)
xpose	数据转置	xpose, clear

以上命令可以通过 help command 查看到具体的命令格式



4.2 案例:拆分与连接数据

4.2.1 横向拆分数据

要完成的任务（任务 4.1）将 mydata 数据分成三个数据文件，分别为学生基本信息文件 student.dta，经济学成绩文件 economy.dta 和数学成绩文件 math.dta。

原始数据文件 mydata.dta

id	name	gender	minority	economy	math
1	John	1	1	40	68
2	Chris	1	2	80	52
3	Jack	0	2	90	76
4	Huang	0	2	43	90
5	Tom	0	3	70	96

6	Han	1	1	53	85
7	Phillip	0	2	85	36
8	Jin	1	2	95	65

将要新生成的三个数据文件如下

student.dta

id	name	gender	minority
1	John	1	1
2	Chris	1	2
3	Jack	0	2
4	Huang	0	2
5	Tom	0	3
6	Han	1	1
7	Phillip	0	2
8	Jin	1	2

economy.dta

id	economy
1	40
2	80
3	90
4	43
5	70
6	53
7	85
8	95

math.dta

id	math
1	68
2	52
3	76
4	90
5	96
6	85
7	36
8	65

横向拆分数据案例的参考操作

***-将 mydata 拆分成学生基本信息数据文件 student-----**
cd d:/mydata/ //在 d: /mydata 文件夹下操作
use mydata, clear //打开第 3 讲已经创建的数据文件
drop economy math //删除 economy 和 math 这两个变量

save student, replace //将删除后的数据集命名为 student，并保存到当前路径

***--将 mydata 拆分成经济学成绩数据文件 economy--**

use mydata, clear //打开第 3 讲已经创建的数据文件

keep id economy //仅保留 id 和 economy 这两个变量在当前数据集中

save economy, replace //将当前数据集重新命名为 economy，并保存到当前路径

***--将 mydata 拆分成数学成绩数据文件 math--**

use mydata, clear

keep id math

save math, replace

4.2.2 纵向拆分数据

要完成的任务（任务 4.2）：将 mydata 数据分成二个数据文件，分别为女生数据集 female.dta 和男生数据集 male.dta。原始数据同上。该任务完成后将要形成的数据如下

female.dta

id	name	gender	minority	economy	math
3	Jack	0	2	90	76
4	Huang	0	2	43	90
7	Phillip	0	2	85	36
5	Tom	0	3	70	96
1	John	1	1	40	68

male.dta

id	name	gender	minority	economy	math
1	John	1	1	40	68
6	Han	1	1	53	85
8	Jin	1	2	95	65
2	Chris	1	2	80	52

纵向拆分数据案例的参考操作

***-----将 mydata 拆分成女生数据集 female-----**

use mydata, clear //打开第 3 讲已经创建的数据文件

keep if gender==0 //仅保留女生的记录在当前数据集中

save female, replace

***-----将 mydata 拆分成男生数据集 male-----**

use mydata, clear

drop if gender==0 //将所有女生的记录全部从当前数据集中删除

save male, replace

4.3 案例：连接数据文件

4.3.1 纵向合并数据

要完成的任务（任务 4.3）：将女生数据集 `female.dta` 和男生数据集 `male.dta` 合并为新的数据集 `mydata1` 原始数据同上。

纵向拆分数据案例的参考操作

```
*-将女生数据集 female 和男生数据集 male 合并为新数据 mydata1-----
use male, clear //打开记录男生信息的数据文件 male
append using female //将记录女生信息的 female 文件追加到当前数据集中
save mydata1, replace
```

4.3.2 横向合并数据

要完成的任务（任务 4.4）：将学生基本信息数据集 `student.dta` 和数学成绩 `math.dta`，经济学成绩 `economy.dta` 合并为新的数据集 `mydata2`。原始数据同上。

横向拆分数据案例的参考操作

```
*-----将学生基本信息和学习成绩合并成新数据 mydata2-----
use economy, clear //打开经济学成绩数据文件
sort id //按学号排序
save economy, replace //重新保存一下
use student, clear //打开学生基本信息数据文件
sort id //按学号排序
merge id using economy //以学号为关联, 将学生的信息和成绩一一对应对接
tab _merge //显示对接情况, 3 表示成功对接, 1 和 2 表示未成功对接
drop _merge //去掉标识对接是否成功变量 _merge
sort id //去掉变量 _merge
save mydata2, replace
use math, clear
sort id
merge id using mydata2 //用学号关联, 将 mydata2 与数学成绩 math 一一对接
drop _merge
save mydata2, replace
```

4.4 数据重整

4.4.1 要掌握的命令

要掌握下面的命令。

命令	命令解释	用法示例
<code>reshape</code>	将数据重整	<code>reshape long inc, i(id) j(yr)</code>
<code>stack</code>	将多列数据转换成一列数据	<code>stack a b c d, into(e f)</code>
<code>xpose</code>	数据转置	<code>xpose, clear</code>

4.4.2 案例:面板数据重整

任务 4.5 数据集 mywide.dta 共有六个变量，其中后四个变量分别为 2003 年和 2004 年的数据成绩和经济学成绩，现要求将数据转化为 mylong.dta 的格式，将年份单独做成变量，数学和经济学成绩则成为两个单独变量。

原始数据 mywide.dta

id	name	math2003	math2004	economy2003	economy2004
1	John	40	13	68	55
2	Chris	80	64	52	87
3	Jack	90	55	76	25
4	Huang	43	60	90	4
5	Tom	70	68	96	42
6	Han	53	10	85	89
7	Phillip	85	61	36	52
8	Jin	95	6	65	84

转换后的数据 mylong.dta

id	name	year	math	economy
1	John	2003	40	68
1	John	2004	13	55
2	Chris	2003	80	52
2	Chris	2004	64	87
3	Jack	2003	90	76
3	Jack	2004	55	25
4	Huang	2003	43	90
4	Huang	2004	60	4
5	Tom	2003	70	96
5	Tom	2004	68	42
6	Han	2003	53	85
6	Han	2004	10	89
7	Phillip	2003	85	36
7	Phillip	2004	61	52
8	Jin	2003	95	65
8	Jin	2004	6	84

重整参考操作

*-----将学习成绩数据集 mywide 变换形式-----

use mywide, clear

reshape long math economy, i(id name) j(year) //数据重整, 宽变长

save mylong, replace

*-----将学习成绩数据集 mylong 变换形式-----

reshape wide

*或者


```
use mylong, clear
reshape wide math economy, i(id name) j(year) //数据重整, 长变宽
save mywide2, replace
```

4.4.3 案例:多列数据转为少数几列

任务 4.6 以下数据集虽然有很多列，但实际上只有一个变量，将该数据复制到 stata 并转化为一项数据。

21	87	23	97	58	3
56	46	9	21	79	17
26	42	59	26	9	29
94	89	35	16	81	62
27	5	80	76	26	89
11	67	10	37	45	48
40	71	57	37	13	22
72	69	10	96	22	93

重整参考操作

```
*-----将多列数据变一列-----
*先将原始数据复制粘贴到 STATA 中
stack var1-var6, into(x) clear
drop _stack
```

4.5 案例:数据转置

任务 4.7：将下面的数据行列互换

原始数据

math.dta

id	math
1	68
2	52
3	76
4	90
5	96
6	85
7	36
8	65

互换后的数据应该为：

newmath.dta

v1	v2	v3	v4	v5	v6	v7	v8
1	2	3	4	5	6	7	8
68	52	76	90	96	85	36	65

转置参考操作

*-----转置-----

use math,clear**xpose, clear**

4.6 字符运算

```

clear
input str15 x
"10*123"
"543*21"
"12*422"
"43532*32134"
"4349*1"
end
gen a=strpos(x,"*")
gen b=substr(x,1,a-1)
gen c=substr(x,a+1,.)

```

1

4.7 复习与作业

drop keep merge append reshape xpose stack

(1) 将第一次要求录入的个人信息数据分别上传到网上，以“i+自己的学号”为文件名保存该数据。然后任意下载 10 位同学的个人信息，合并成一个文件，并加标注和说明信息。改名为“we+自己的学号”后保持。

学生基本信息调查表				
学号 id	姓名 name	性别 gender	年龄 age	家乡省份 province

(2) 将自己家乡省份 2000-2005 年各年的 GDP、人口、农村居民人均纯收入数据收集并做成数据集，文件名为 myhm+学号。上传到网络，各自下载 10 位同学的该文件，并将其合并为一个数据集，文件名为“ourhm+学号”。

省调查表			
学号 id	家乡省份 province	省历年 gdp2000-gdp2005	农村居民人均纯收入 income2000-2005

(3) 再将第一个题和第二题中形成的两个合并文件进一步合并为一个文件。(假设你是该省的联络人，这种表格是有一定实际意义的)。文件名为“ourpv+学号”。

(4) 将第二题中的数据变换结构如下。文件名为 mylg+学号

id	province	year	gdp	pop	income
1	北京	1980			
1	北京	1981			
1		...			
2	上海	1980			
2	上海	1981			
2		...			

5 函数与运算符

5.1 运算符 exp

STATA 共有四种运算，分别是代数运算、字符运算、关系运算和逻辑运算。

运算符一览表

代数运算		逻辑运算		比较关系	
+	加	!	不	>	大于
-	减	~	不	<	小于
*	乘		或	>=	不小于
/	除	&	和	<=	不大于
^	指数			==	等于
sqrt()	开方			!=	不等于
~=	不等于			~=	不等于
+	字符相加				

运算的优先序：! (或~)，^，-（负号），/，*，-（减），+，!=(或~=),>,<,<=,>=,==,&,&|
当忘记或者无法确定优先序的时候，最好用括号将优先序表达出来，在最里层括号中的表示式将被优先执行。

5.1.1 代数运算

包括加 (+)、减 (-)、乘 (*)、除 (/)，幂 (^) 和负数 (-)，当遇到缺失值或者运算不可行时（比如除数为零）均会得到缺失值。

例：求下式的值，若 $x=4, y=2$ ，显然，经过心算，应该为：-1

$$\frac{x + y^{x-y}}{xy}$$

. di -(4+2^(4-2))/(2*4) //di 是 display 命令的略写，表示显示结果

. di 4-2 //输出 2

. di 3*5 //输出 15

. di 8/2 //8 除以 2，输出 4

. di 2^3 //2 的立方，输出 8

. di -(2+3^(2-3))/sqrt(2*3) //括号运算优先，想一想，结果应为多少？

实际上，更多的情形是两个或多个变量的直接运算。比如，将进口车的价格都增加 100 元（可能是关税），而国产车不变。

. sysuse auto, clear

. gen nprice=price+foreign*100 /*由于 foreign 的取值为 0 和 1，因此只有进口车的价格增加 100 元*/

. list nprice price foreign

5.1.2 字符运算

加 (+) 号同样可用于字符运算，当加号出现在两个字符之间时，两个字符将被连成一个字符。比如把“我爱”“STATA”合并在一起，命令为：

```
. scalar a="我爱"+"STATA"    //要特别注意，引号必须是半角和英文模式
. scalar list a                //scalar 命令将两个字符运算后的结果赋于 a，然后显示 a
. scalar a=2+"3"              //注意到：字符与数值不能直接相加，显示类型不匹配
                                type mismatch
                                r(109);                //双击天蓝色的错误代码，可以进入错误解释。
```

5.1.3 关系运算

关系运算包括大于、小于、等于；不等于、不小于、不大于等多种比较关系。特别要注意到 STATA 中的等于符号为“==”，是两个等号连写在一起，不同于赋值时用的单个等号“=”。

```
. di 3<5      //输出结果为 1，意味着 3 小于 5 为真
. di 3>5      //输出的结果为 0，意味着 3 大于 5 为假
. di 3==4     //输出的结果为 0，意味着 3 不等于 4
```

当数据中含有缺失值的时候需要特别小心，因为系统缺失值大于任何一个数据，利用这一点，我们可以使用条件语句排除缺失值。

任务：将年龄分组为 65 岁以下和 65 岁及以上两组，缺失值显然不能包括在任何一组中。

age
38
.
65
42
18
80

```
. clear
. edit
```

将上述数据复制到 STATA 中，然后退出数据编辑器。

```
. gen agegrp1=(age>=65)
```

生成的数据中，将缺失值视为 65 岁以上分在了高龄组，这是错误的

```
. gen agegrp2=(age>=65) if age<.
```

生成的数据中，将缺失值排除在外，正确！这一命令常被用于生成虚拟变量。

```
. gen agegrp3=(age==65) if age<. //仅判断是否恰好为 65 岁
```

```
. list //比较 agegrp1、 agegrp2 和 agegrp3 的差异，体会 if age<.的作用。
```

age	Agegrp1	Agegrp2	Agegrp3
38	0	0	0
.	1	.	.
65	1	1	1
42	0	0	0

18	0	0	0
80	1	1	0

5.1.4 逻辑运算

逻辑运算包括非 (!)，和 (&)、或 (|) 三种，主要用于条件语句中。

例：列示出价格大于 10000 元的任何车，或者小于 4000 元的国产车。

```
. sysuse auto, clear
```

```
. list price foreign if price>10000 | price<4000 & forei==0
```

在 STATA 中，和 (&) 优先于或 (|)，因此上述命令与下面的命令等价：

```
. list price foreign if price>10000 | (price<4000 & forei==0)
```

试一试下面的命令，这里列示的是国产车中价格高于 10000 元或者低于 4000 元的车。

```
. list price foreign if (price>10000 | price<4000) & forei==0
```

5.2 函数概览 function

函数只不过是一些编号的小程序，这些小程序会对数据按一定的规则进行处理，之后报告结果。实际上，谁也记不住这么多函数，因此，首先要学会查找函数的帮助，当记不住的时候，随时去查寻帮助。记住下面的命令才是最关键的。

```
. help function
```

Type of function

See help

Mathematical functions

math functions

Probability distributions and
density functions

density functions

Random-number functions

random-number functions

String functions

string functions

Programming functions

programming functions

Date functions

date functions

Time-series functions

time-series functions

Matrix functions

matrix functions

弹出来的对话框告诉我们，STATA 包括八类函数，分别是数学函数，分布函数，随机数函数，字符函数，程序函数，日期函数，时间序列函数和矩阵函数。本章主要介绍数学函数和字符函数，其他函数将在后面相应的章节介绍。

常用函数一览表

	函数	含义	举例
数值 型函 数	abs(x)	绝对值	abs(-9)=9
	comb(n, k)	从 n 中取 k 个的组合	comb(10, 2)=45
	exp(x)	指数	exp(0)=1
	fill()	自动填充数据	
	int(x)	取整	int(5.6) = 5, int(-5.2) = -5.
	ln(x)	对数	ln(1)=0

	log10(x)	以10为底的对数	log10(1000)=3
	mod(x, y)	= x - y*int(x/y)	mod(9, 2)=1
	round(x)	四舍五入	round(5.6)=6
	sqrt(x)	开方	sqrt(16)=4
	sum(x)	求和	
随机函数	uniform()	均匀分布随机数	第10讲将介绍
	invnormal(uniform())	标准正态分布随机数	第11讲将介绍
字符函数	real(s)	字符型转化为数值型	
	string(n)	数值型转化为字符型	
	substr(s, n1, n2)	从S的第n1个字符开始，截取n2个字符	Substr(“this”, 2, 2)=is
	word(s, n)	返回s的第n个字符	Work(“this”, 3)=i
系统变量	_n	当前观察值的序号	
	_N	共有多少观察值	
	_pi	π 为圆周率	3.14159

5.3 数学函数 math functions

5.3.1 三角函数，指数和对数函数

数学函数可以直接对数据进行运算，也可以对变量进行运算。

```
. di sqrt(4)           //开方，输出 2
. di sqrt(6+3)         //先相加，再开方，输出 3
. di abs(-100)         //求绝对值，输出 100
. di exp(1)            //表示  $e^1$ ，输出 2.7182818
. di ln(exp(2))        //先求  $e^2$ ，再取对数，得到 2
. di _pi               //_pi 为圆周率，得到 3.1415927
. di cos(_pi)          //_pi 的余弦值，得到 -1
```

数学函数可以直接对数据进行运算，也可以对变量进行运算。

对变量的操作：

```
clear
set obs 5           //设定 5 个观察值
gen x=_n            //生成新变量 x,取值为 1, 2, 3, 4, 5
gen y1=exp(x)       //取指数
gen y2=ln(x)        //取对数
gen y3=sin(exp(x)) + cos(ln(x)) //取对数
l                   //显示刚生成的数据
```

5.3.2 取整和四舍五入

取整

```
. di int(3.49)      //int()取整，不论后面的小数是什么，只取小数点前的数值
. di int(3.51)      //输出 3
. di int(-3.49)     //输出-3
. di int(-3.51)     //输出-3
```

四舍五入

```
. di round(3.49)    //round()取整，四舍五入，结果为 3
. di round(3.51)    //四舍五入，结果为 4
. di round(-3.49)   //四舍五入，结果为-3
. di round(-3.51)   //四舍五入到个位数，结果为-4

. di round(3.345,.1) //四舍五入到十分位，结果为 3.3
. di round(3.351,.1) //四舍五入到十分位，结果为 3.4
. di round(3.345,.01) //四舍五入到百分位，结果为 3.35
. di round(3.351,.01) //四舍五入到百分位，结果为 3.35
. di round(335.1,10) //四舍五入到十位，结果为 340
```

对变量的操作

```
. sysuse auto, clear
. gen nprice=price/10000 //将价格变到以万为单位
. gen nprice2=round(nprice,0.01) //四舍五入到百分位
. list nprice*           //比较结果
```

5.3.3 求和及求均值 gen 和 egen

```
clear
set obs 5
gen x=_n //生成新变量 x, x 的取值从 1 到 5
gen y=sum(x) //求列累积和
egen z=sum(x) //求列总和，注意比较 y 和 Z 的不同
egen r=rsum(x y z) //求 x+y+z 总和
egen havg = rowmean(x y z) //求 havg=(a+b+c)/3
egen hsd = rowsd(x y z) //求 a、b 和 c 的方差
egen rmin = rowmin(x y z) //求 x y z 这三个变量的最小值
egen rmax = rowmax(x y z) //求 x y z 这三个变量的最大值
list //注意比较 y 和 z 的不同。
```

```
egen avgx=mean(x) //求列均值
egen medx=median(x) //求列中值
egen stdx = std(x) //求变异系数  $cv_i=(x_i-mx)/s$ , 注意  $s=\sqrt{\sum (x_i-mx)^2/(n-1)}$ 
replace y=3 in 3
egen byte dxy = diff(x y) //当 x 与 y 相等时，differ 取 0，若不相等为 1
更多关于 egen 命令的用法将参考帮助：help egen
```


5.3.4 其他

```

sysuse auto, clear
egen rmpg = rank(mpg) //求 mpg 的次序
sort rmpg
list mpg rmpg //列示结果
egen highrep78 = anyvalue(rep78), v(3/5) /*若 rep78 不为 3、4 或 5,
                                          则为缺失值*/

list rep78 highrep78

clear
input a b
1 0
0 0
1 1
0 1
0 0
1 .
. 0
end
egen ab=group(a b) /*按 a 和 b 来进行交叉分组, a=0,b=0 为第一组
                   , ...,a=1,b=1 为第四组, 缺失值不参与分组*/
egen ab2=group(a b),missing //将缺失的组当作另外的一组
l //显示分组结果
a b ab ab2
-----
1. 1 0 3 3
2. 0 0 1 1
3. 1 1 4 4
4. 0 1 2 2
5. 0 0 1 1
-----
6. 1 . . 5
7. . 0 . 6

clear
set obs 100 //设定 100 个观察值
gen age=_n //生成一个假设的年龄变量 age, 依次取 1, 2, ..., 100
recode age (min/30=1) (30/60=2) (60/max=3),gen(agegrp) /*生成新的分组
                                                         变量 agegrp,当年龄 age 在 30 及以下时取值
                                                         为 1, 30 到 60 为 2, 60 以上为 3*/

```

5.4 字符函数 string functions

任务：将美国汽车数据中汽车商标变量值简化为取前三个字母，得到一个新的变量 `make3`

```
sysuse auto, clear
```

```
gen str3 make3=substr(make,1,3) //取变量 make 的前三个字符赋给 make3
```

```
list make*
```

任务：下表的数据是一个多选题，请把这道多选题转化为四个单选题，例如变量 `a` 表示“你今天早餐吃的是：1 稀饭 2 馒头 3 大饼 4 豆腐脑”。变量 `a` 不容易处理，可以用以下命令转化为四个 0-1 变量，分别为 `na1`=是否吃稀饭，`na2`=是否吃馒头…

a
2
2
2、1、3
1、2、4
4、2、1
1、2
2
1、2

```
gen na1=strpos(a, "1")!=0
```

/*strpos(s1,s2)返回字符 s2 在 s1 中的位置,如果在 s1 中找不到 s2，则返回 0。上述命令是实际上有两步，先得 0 或非零，然后判断转化为 0-1。

```
gen na2=strpos(a, "2")!=0
```

```
gen na3=strpos(a, "3")!=0
```

```
gen na4=strpos(a, "4")!=0
```

```
list
```

	a	na1	na2	na3	na4
1.	2	0	1	0	0
2.	2	0	1	0	0
3.	1、2、3	1	1	1	0
4.	1、2、4	1	1	0	1
5.	1、2、4	1	1	0	1
6.	1、2	1	1	0	0
7.	2	0	1	0	0
8.	1、2	1	1	0	0

```
sysuse auto, clear
```

```
gen name2 = word(make,2) //新变量 name2，取值为 name 的第二个单词，  
注意是单词而不是字母，以空格为分隔
```

```
list n*
```

```
di word("this is a dog",4) //显示第四个字母 dog
```

5.5 分类操作 by

```
clear
```

```
edit
```

*将下表复制粘贴到 STATA 数据编辑器中，注意粘贴时把光标停在第一格。

x	y
1	1.1
1	1.2
1	1.3
2	2.1
2	2.2

```
gen n=_n      //生成一个新变量 n=1, 2, 3, 4, 5
```

```
gen N=_N      //生成一个新变量 N=5, 5, 5, 5, 5
```

```
gen z=y[1]    //生成一个新变量 z=y 的第一个观察值
```

```
list
      x      y      n      N      z
```

```
1.  1      1.1      1      5      1.1
```

```
2.  1      1.2      2      5      1.1
```

```
3.  1      1.3      3      5      1.1
```

```
4.  2      2.1      4      5      1.1
```

```
5.  2      2.2      5      5      1.1
```

```
by x, sort: gen n1=_n    //注意到 n1 与 n 的不同，n1 按 x 分类进行操作
```

```
by x, sort: gen N1=_N
```

```
by x, sort: gen z1=y[1]
```

```
list
```

```
      x      y      n      N      z      n1      N1      z1
```

```
1.  1      1.1      1      5      1.1      1      3      1.1
```

```
2.  1      1.2      2      5      1.1      2      3      1.1
```

```
3.  1      1.3      3      5      1.1      3      3      1.1
```

```
4.  2      2.1      4      5      1.1      1      2      2.1
```

```
5.  2      2.2      5      5      1.1      2      2      2.1
```

任务：下列数据为家庭成员数据 family.dta，其中 hhid 为家庭编码，age 为家庭成员的年龄。将下表数据复制到 STATA，然后另存为 family.dta

hhid	age
1	86
1	42
1	36

1	57
1	28
2	42
2	5
2	40

要求：(1) 生成一个新变量 `hhsiz`，该变量表示共有多少个家庭成员。(2) 给每个家庭成员一个编码 `id`。如第一个家庭的第一个成员编码为 11；(3) 按家庭生成一个全家成员平均年龄值 `mage`。(4) 对每个家庭，分别按年龄大小排序，然后生成一个家庭成员代码，即家庭内年龄最小的成员代码为 1，年龄最大的家庭成员，代码为 `nid`。

最后需要生成的数据集如下：

hhid	age	hhsiz	id	mage	nid
1	28	5	15	49.8	1
1	36	5	13	49.8	2
1	42	5	12	49.8	3
1	57	5	14	49.8	4
1	86	5	11	49.8	5
2	5	3	22	29	1
2	40	3	23	29	2
2	42	3	21	29	3

请自己先思考，再参考如下操作：

将上表数据复制粘贴到 STATA 数据编辑器，然后执行下面的命令

use family, clear

by hhid, sort : gen hhsiz = _N //得到家庭规模 `hhsiz`

by hhid, sort : gen id = _n + hhid * 10 //为家庭成员编码

by hhid, sort : egen mage = mean(age) //求平均年龄

sort hhid age //按户排序，在每个户内按年龄大小排序

by hhid : gen nid = _n //在户内按年龄大小为家庭成员编码

	hhid	age	hhsiz	id	mage	nid
1.	1	28	5	11	49.8	1
2.	1	36	5	12	49.8	2
3.	1	42	5	13	49.8	3
4.	1	57	5	14	49.8	4
5.	1	86	5	15	49.8	5
6.	2	5	3	21	29	1
7.	2	40	3	22	29	2
8.	2	42	3	23	29	3

另一个例子：

```

use family, clear
bysort hhid (age): gen nid1=_n //括号中的变量 age 只排序，不参与分组。
bysort hhid age: gen nid2=_n // hhid 和 age 都既用来参与排序也分组
list //比较上面两个命令得到的不同结果

```

	hhid	age	nid1	nid2
1.	1	28	1	1
2.	1	36	2	1
3.	1	42	3	1
4.	1	57	4	1
5.	1	86	5	1
6.	2	5	1	1
7.	2	40	2	1
8.	2	42	3	1

```

webuse stan2, clear
expand 2 if transplant //将 transplant==1 的观察值再复制一个
sort id
by id: generate byte posttran = (_n==2) /*生成一个新变量 posttran，使得
                                         对同一个人，第一个观察值取 0，第
                                         二个观察值取 1*/
by id: generate t1 = stime if _n==_N /*生成新变量 t1，使得在同一个 id 下，
                                         对第二期取值为 stime，否则为 “.”
gsort -id

```

6 程序

6.1 标准的程序文件格式

我们已经多次强烈建议大家尽量用程序来完成自己的工作。在写程序时，有一些经验写法，遵循这些写法将会提高工作效率。

```

=====begin=====
clear          //相当于让 STATA 处于初始状态，清除所有使用过的痕迹
version 9      //由于不同版本命令等略有不同，因此最好事先指明版本号
cd d:/stata9   //设定路径，将数据、程序和输出结果文件均存入该文件夹
capture log close /*如某结果输出文件已被打开，则关闭之，
                  若没有，则忽略该命令*/
log using myfile, replace //将运行结果存到一个输出文件 myfile 中
set more off    //在程序执行过程中，不要因为结果窗口屏幕已满而停下来
log off        //暂时关闭结果记录功能，以下的执行和结果均不记录

*下面开始写完成特定任务的命令，如
sysuse auto, clear
sum
log on          //打开结果记录功能，以下命令和结果记录
tab forei

log close      //关闭结果输出文件，在前面设定的文件目录中可以找到。
=====end=====

```

Log 命令记录所有已执行命令或执行结果，结果文件的存贮类型有两种，一种后缀名为*.smcl，一种为*.txt，如果不指明为 txt，默认为*.smcl。后面的 replace 选项用于覆盖原来的同名结果文件。

6.2 创造自己的命令：与 STATA 互致问候

在写程序的过程中，如果遇到要反复调用同一段代码时，明智的选择是将这段代码写成子程序，然后直接调用子程序即可。

试一试，在命令窗口输入“hello”，会出现什么结果？

```
hello
```

```
unrecognized command
```

红字告诉你，STATA 无法认识这个命令。

我们可以自己创建一个叫 hello 的命令，当你运行这个命令的时候，STATA 就会向你问好“你好，老兄！”，下面是程序。

在程序编辑器内键入以下内容并执行之

```

=====begin=====
capture program drop hello

```

```

program hello
display “你好,老兄”    //请注意引号一定要在中文状态下输入
end

```

```

*=====end=====

```

* 然后在命令窗口键入

```
hello
```

* STATA 将在结果窗口显示出

```
*你好,老兄
```

*我们来去做做其他工作，然后再次调用该命令 **hello**，看灵还是不灵。

```
sysuse auto, clear
```

```
list make price in 1/5
```

```
des
```

```
hello
```

这个命令居然像 **describ** 或者 **list** 一样！是的，这是一个名为 **hello** 的命令，一旦该程序被读入内存（即被执行一次），只要你不退出 **stata** 或者删除该程序 (**program drop hello**)，他就随时待命，你可以像用其他命令一样来使用它。

那么如果我们退出 **stata** 后还想执行这个程序，怎么办呢？你可以把两条红线之间的这个文件存起来，最好是存在 **STATA** 的默认路径下，文件名为 **hello.do**（注意后缀为 **.do**），**退出 STATA 并重启**（绝对必要，为什么？），在命令窗口输入命令

```
do hello
```

则输出结果：

```
“你好,老兄”
```

这就是最简单的程序文件，如果你想每次打开 **STATA** 后都先向 **STATA** 打个招呼，我们可以将这个程序写好后存入 **stata/ado/h** 下面，文件名为 **hello.ado**（注意后缀为 **.ado**）。则每次开机后就可以直接输入：

```
hello
```

STATA 也将回应你：

```
“你好,老兄”
```

比较不爽的是，它就像鹦鹉，只会说“你好，老兄”，能不能说点别的啊？*/

```

*=====begin=====

```

```
capture program drop hello
```

```
program hello
```

```
display as error “`1’”    /*请注意左单引号位于标准键盘左上角，
                           右单引号在回车键旁*/
```

```
end
```

```

*=====end=====

```

*这时，我们想让他说什么就在 **hello** 后面说什么。

```
hello 吃了吗？
```

```
hello 吃了，你呢？
```

```
hello 我也吃了
```

*在例子中，有几个陌生的命令，如
***capture program drop** *programname*
***program** *programname*
 *`1’
 *请体会他们的用法！

自创命令的一般结构

```
capture program drop progrname
program progrname
    command
end
```

capture 命令的用法

当你试图再次运行该程序时，系统马上会提醒

```
program hello
hello already defined
```

这是因为 *hello* 这个程序已经存在的，要重新录入，需要先将原来的程序删除

```
*=====begin=====
program drop hello
program hello
display “你好,老兄”
end
hello
*=====end=====
```

可是，当我们将该程序保存并退出 *stata*，然后重新打开 *stata* 并试图再次运行该程序时，错误又出现了，

```
hello not found
```

这是因为 *stata* 按顺序来执行命令，先执行 **program drop hello**，然而 *hello* 却不在内存中。因此，标准的程序中通常要加上一个灵活选择性命令 **capture**，意思是，如果有 *hello* 这个命令，则执行删除命令，如果没有，则跳过这条命令，执行下一条命令。*/

```
*=====begin=====
capture program drop hello
program hello
display “你好,老兄”
end
hello
*=====end=====
```


6.3 暂元 Macros: local/global

暂元是程序中的临时变量，分为暂元名和暂元内容两部分，类似于变量名和变量值。

```

=====begin=====
sysuse auto, clear
list price length weight in 1/5
local v3 "price length weight" //将 price length weight 这组字符赋给暂元名 v3
list `v3' in 1/5              /*等价于 list price length weight in 1/5。注意 `v3' 中，左边不是单引
                              号，而是标准键盘上第二排最左边的哪个撇号。*/
local cmd "list"              //将 list 这组字符赋给暂元名 cmd
`cmd' `v3' in 1/5              //等价于 list price length weight in 1/5
local pre "pri"               //将 pri 这组字符赋给暂元名 pre
local suf "ce"                //将 ce 这组字符赋给暂元名 suf
`cmd' `pre' `suf'              //等价于 list price
tab rep`=26*3'                //等价于 tab rep78，注意到 26*3=78

```

*类似地可以使用 **global** 来定义暂元

```

sysuse auto, clear
list price length weight in 1/5
global v3 "price length weight" //将 price length weight 这组字符赋给暂元名 v3
list $v3 in 1/5                  //等价于 list price length weight in 1/5。
global cmd "list"                //将 list 这组字符赋给暂元名 cmd
$cmd $v3 in 1/5                  //等价于 list price length weight in 1/5

```

```

=====end=====

```

暂元定义	等价于
global a "myvar" gen \$a=oldvar gen a=oldvar	gen myvar=oldvar gen a=oldvar
local a "myvar" gen `a'=oldvar gen a=oldvar	gen myvar=oldvar gen a=oldvar
global a "newvar" global i=2 gen \$a\$i=oldvar	gen newvar2=oldvar
local a "newvar" local i=2 gen `a'`i'=oldvar	gen newvar2=oldvar

<pre>global b1 "newvar" global i=1 gen \${b\$i}=oldvar local b1 "newvar" local i=1 gen `b`I'=oldvar global b1 "newvar" global a "b" global i=1 gen \${a\$i}=oldvar local b1 "newvar" local a "b" local i=1 gen ``a``i'=oldvar</pre>	<pre>gen newvar=oldvar gen newvar=oldvar gen newvar=oldvar gen newvar=oldvar gen newvar=oldvar</pre>
--	--

Global 与 local 的区别

capt prog drop myprog

```
program myprog //定义程序 myprog
local i="主程序局域" //定义局域暂元 i
global j="主程序全局" //定义全局暂元 j
di as txt "`i'" //调用局域暂元 i，这里是显示暂元名 i 中的内容“主程序局域”，绿色
di as txt "$j" //调用全局暂元 j，这里是显示暂元名 j 中的内容“主程序全局”，绿色
mysub //执行子程序体 mysub
di as error "`i'" //再次调用局域暂元 i，显示暂元名 i 中的内容“主程序局域”
di as error "$j" //再次调用全局暂元 j，显示“子程序全局暂元”，红色
end
```

capt prog drop mysub

```
prog mysub //定义子程序 myprog
local i="子程序局域" //定义子程序中的局域暂元 i
global j="子程序全局" //定义子程序中的全局暂元 j
di as result "`i'" //调用局域暂元 i，这里是显示暂元名 i 中的内容“子程序局域暂元”
di as result "$j" //调用全局暂元 j，这里是显示暂元名 j 中的内容“子程序全局暂元”
end
myprog //执行主程序体 myprog
```

得到如下结果

主程序局域

主程序全局

子程序局域暂元

子程序全局暂元

主程序局域

子程序全局暂元

在上述程序的执行过程中，首先显示主程序所定义的暂元内容。然后调用子程序，显示子程序中的命令规定的暂元内容。再回到主程序，此时在主程序中用 `local` 所定义的暂元并没有因为子程序的重新定义或调用而改变，仍然保持着在主程序中所定义的内容，即“主程序局部”，但是由 `global` 所定义的全局暂元却发生了改变，变为由子程序所更新定义后的“子程序全局暂元”。

6.4 自带命令参数

在前面的例子中，我们不仅创建了一个自己的命令 `hello`，而且还可以让 `hello` 想说什么就说什么，即自带命令参数，实际上，命令参数可以是字符，数值，也可以是变量，矩阵，文件等任何内容。命令参数也不仅限于一个，可以是多个。

```
captu prog drop listargs
prog listargs
di "第一个参数为:  `1' "
di "第二个参数为:  `2' "
di "第三个参数为:  `3' "
di "第四个参数为:  `4' "
end

listargs
listargs I love stata
listargs "I love stata"
local i "I love stata"
listargs `i'

listargs this is a test
listargs "this is a test"
listargs "this is" "a stata test"
```

另一个办法是采用参数声明的方式。

```
captu prog drop listargs
prog listargs
args a b c d
di "第一个参数为:  `a' "
di "第二个参数为:  `b' "
di "第三个参数为:  `c' "
di "第四个参数为:  `d' "
end

listargs this is a test
listargs "this is a test"
```

6.5 scalar 标量

```

scalar a=2           //赋予标量 a 的值为 2
dis a+2              //a+2=2+2=4
scalar b=a+3         //b=a+3=2+3=5
di b                 //结果窗口显示出： 5
scalar s="hello"     //标量也可以为字符型
di s                 //结果窗口显示出： hello

clear
set obs 3
input a
1
3
5
end
scalar b=3
gen v1=a*b           //回忆线性代数，该命令相当于对 a 的每个值乘 b(即 3)
list
input b
2
3
4
end

gen v2=a*b           //注意 v1 与 v2 这两个命令完全相同
list                //v1 与 v2 结果不同，表明 STATA 优先进行变量相乘

```

```

gen v3=a*scalar(b)   //当 b 即是变量名又是标量名时，标量相乘时要指明标题
list

```

说明：当变量名和标量名相同时，STATA 将优先使用变量名，如果要强制使用标题，需要用 `scalar(varlist)`

```

sysuse auto, clear
sum price
return list //计算结果同时被存入到如下的标量中
scalars:
r(N) = 74
r(sum_w) = 74
r(mean) = 6165.256756756757
r(Var) = 8699525.974268789
r(sd) = 2949.495884768919
r(min) = 3291
r(max) = 15906
r(sum) = 456229

```

```
local sum=r(sum)    //将总和存入一个叫 sum 的暂元中
di `sum'
```

6.6 临时变量和临时数据文件:tempvar 和 tempfile

在我们的数据处理过程中，会产生一些中间的临时性变量，如果随着退出 STATA，这些临时的中间性变量能自动消失，既不会破坏原始数据，也不会因为这些干扰变量妨碍我们查看结果性的变量。

```
Clear
clear
set obs 5
tempvar x y    //指明临时变量名为 x 和 y
gen `x'=_n     //生成临时变量 `x'，取值为 1, 2, 3, 4, 5
gen `y'=_N     //生成临时变量 `y'，取值为 5, 5, 5, 5, 5
edit           //查看数据编辑器，却并没有变量
gen z=`x'+`y'  //将两个临时变量相加，得到新变量 z
edit           //查看数据库，只有新变量 z
1             //显示数据，注意临时变量的变量名为 __000000  __000001
```

在程序的运行过程中，我们需要破坏内存中的数据文件来获得预期结果，但我们并不希望改变内存的数据，和临时变量一样，我们可以依赖临时数据文件。一般使用 preserve...restore，这两个命令之间的命令不会对数据造成任何破坏，但是一个可能的风险是：他人在运行该文件时强行中止，来不及执行 restore。这会使得数据仍然被破坏。而临时数据文件则不存在这样的风险。

```
*=====begin=====
use auto, clear
preserve           //表明下面的操作将不破坏当前数据中的文件
keep price weight //仅保留 price 和 weight 这两个变量
save master, replace //保存新数据文件 master，该数据包含两个变量
drop weight       //再去掉变量 weight，当前数据中只剩一个变量 weight
save part1, replace //保存新数据文件 part1，该数据包含一个变量
use master, clear
drop price
rename weight price
append using part1
erase master.dta
erase part1.dta
restore           //回到 preserve 命令之前的数据集，即 auto
*=====end=====
```

/*这是一个坏程序，因为中间性数据文件 master 和 part 并不是我们想要的，即使最后有 erase 命令将其删除，一旦在此前强行退出，仍会导致这两个不必要的文件占用空间并造成混乱。

```
*=====begin=====
preserve
```

```

keep price weight
tempfile master part1
save "`master'"
drop weight
save "`part1'"
use "`master'", clear
drop price
rename weight price
append using "`part1'"
restore

```

/*在这种情形下，无论是出现运行错误还是中途中止，都不会破坏原数据文件，也不会产生垃圾文件。*/

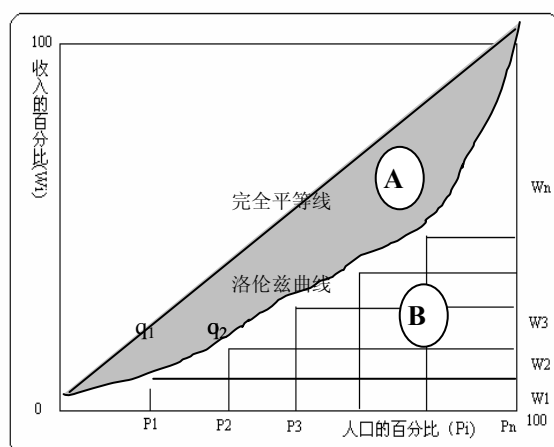
```

=====end=====

```

6.7 基尼系数命令的创建案例（选学内容）

6.7.1 背景知识：基尼系数



基尼系数是通过计算洛伦茨曲线图中洛伦茨曲线与对角线之间的面积 A 以及对角线右下方的直角三角形面积 (A+B)，将这两块面积相除而求得。即基尼系数=A/ (A+B)。在基尼系数的计算上存在许多公式和算法，一种较直观简便的计算方法是：

假定样本人口可以分成 n 组，设 w_i 、 m_i 和 p_i 分别代表第 i 组的收入份额、平均人均收入和人口频数 ($i=1,2,\dots,n$)，对全部样本按人均收入 (m_i) 由小到大排序后，基尼系数

(G) 可由公式 (1) 计算出来：

$$G = 1 - \sum_{i=1}^n 2B_i = 1 - \sum_{i=1}^n p_i(2Q_i - w_i) \quad (1)$$

$$Q_i = \sum_{k=1}^i w_k$$

Q_i 为从 1 到 i 的累积收入比重。B 为洛伦茨曲线右下方的面积。 p_i 、 w_i 从 1 到 n 的和为 1。
任务：试编写一个计算 gini 系数的命令，并计算湖北农户的基尼系数，数据 hldata97.dta，部分数据如下表（其中 hhid 为户代码，inc 为家庭纯收入，hhsz 为家庭人口数）。

6.7.2 粗糙且只能使用一次的程序

```

=====begin=====
cd c:\lex3
//定位路径
中国人民大学 陈传波
chriscb@126.com

```

```

use hb97, clear           //打开数据
gen m=inc/hhsize          //m 为人均纯收入
sort m                   //按人均纯收入由小到大排序
egen tp=sum(hhsize)       //求总人口 tp
egen tinc=sum(inc)        //求总收入 tinc
gen p=hhsize/tp           //求每个家庭人口占总人口的比例 p
gen w=inc/tinc            //求家庭纯收入占总收入的比例 w
gen q=sum(w)              //求按人均纯收入排序后的累积收入比例
gen gini=1-sum(p*(2*q-w)) //计算出基尼系数（即 gini 变量的最后一个数）
dis gini[_N]              //显示基尼系数
*注意到 gen 和 egen 在求和方面的不同，前者求出累积和，后者求出总和

*=====end=====

*运行结果为
*. 26685518

```

*如果为上面的这段程序加上程序头和程序尾，并取名为 gini，则该程序将能重复执行。

```

*=====begin=====

capture program drop gini //如果 gini 命令已存在，则删除，否则跳过该步
program gini              //定义命令名为 gini
gen m=inc/hhsize          //m 为人均纯收入
sort m                   //按人均纯收入由小到大排序
egen tp=sum(hhsize)       //求总人口 tp
egen tinc=sum(inc)        //求总收入 tinc
gen p=hhsize/tp           //求每个家庭人口占总人口的比例 p
gen w=inc/tinc            //求家庭纯收入占总收入的比例 w
gen q=sum(w)              //求按人均纯收入排序后的累积收入比例
gen gini=1-sum(p*(2*q-w)) //计算出基尼系数（即 gini 变量的最后一个数）
dis gini[_N]              //显示基尼系数
end

*=====end=====

use hb97,clear
gini //执行该命令，得到计算结果
*. 26685518

```

6.7.3 变量声明: args

上面的命令只能用于计算家庭纯收入的基尼系数，如果我们想计算现金收入或者食品消费的基尼系数？就必须重新修改命令，至少需要将上面程序中的 inc 替换为 cash 或者 food，有没有一种办法，使得我们不必修改程序，每次使用时，只要在 gini 命令之后带上要计算的变量即可？*/

```

*=====begin=====

capture program drop gini //如果 gini 命令已存在，则删除，否则跳过该步
program gini              //定义命令名为 gini
args inc hhs              //规定命令 gini 后要带两个变量 inc 与 hhs
egen tinc=sum(`inc')      //计算 gini 命令后第一个变量（收入支出等）的总和

```

```

egen tp=sum('hhs')           //计算 gini 命令后第二个变量（人口）的总和
gen m='inc'/'hhs'             //计算人均水平值 m
sort m                       //按人均水平值排序
gen gini=1-sum('hhs'/tp*(2*sum('inc'/tinc)-'inc'/tinc)) //计算基尼系数
dis gini[_N]                 //显示基尼系数
end                           //程序结束

*=====end=====

use hb97,clear
gini cash hhs               //计算现金收入的基尼系数
*.35139242

```

6.7.4 用`1'`2' 等来声明命令后的参数

STATA 也提供了另一种选择，不需要使用 args 语句，请看下面的做法*/

```

*=====begin=====

capture program drop gini //如果 gini 命令已存在，则删除，否则跳过该步
program gini              //定义命令名为 gini
egen tinc=sum('1')        //计算 gini 命令后第一个变量（收入支出等）的总和
egen tp=sum('2')          //计算 gini 命令后第二个变量（人口）的总和
gen m='1'/'2'             //计算人均水平值 m
sort m                   //按人均水平值排序
gen gini=1-sum('2'/tp*(2*sum('1'/tinc)-'1'/tinc)) //计算基尼系数
dis gini[_N]             //显示基尼系数
end                       //程序结束

*=====end=====

```

/*在上面的程序中，我们用`1'和`2' 代替了前两段程序中的 inc 和 hhs。这一变换使得 gini 程序可以用于计算其他变量的不均等程度，如 */

```

use hb97,clear
gini inc hhs               //计算现金收入的基尼系数
*.26685518

use hb97,clear
gini cash hhs              //计算现金收入的基尼系数
*.35139242

*gini food hhs             //计算食物消费的基尼系数
*tinc already defined
*出错了！STATA 说 tinc 已经定义过了，为什么会这样，又该怎么办？

```

6.7.5 使用暂元

注意到在上面的命令执行后，内存数据中冒出了许多中间变量，如 tp tinc m, gini 等等。正是这些变量阻碍了对该命令的扩展运用。如果每执行一个命令都会生成这么多变量，将会对原始数据带来多沉重的负担！想想 STATA 所提供的命令一般都不破坏原数据，我们能否也不破坏原数据呢？答案是：使用暂元（临时变量） */

```

*=====begin=====

```



```

capture program drop gini
program gini
tempvar tinc tp m gini //设定 tinc tp m gini 四个变量为临时变量
egen `tinc'=sum(`1') //生成总收入，将总收入数据暂存在临时变量'tinc'中
egen `tp'=sum(`2')
gen `m'=`1'/`2'
sort `m'
gen `gini'=1-sum(`2'/`tp'*(2*sum(`1'/`tinc')-`1'/`tinc'))
dis `gini'[_N] //显示基尼系数
end

*=====end=====

use hb97,clear
gini inc hhsz //计算现金收入的基尼系数
*.26685518
gini cash hhsz //计算现金收入的基尼系数
*.35139242
gini food hhsz //计算食物消费的基尼系数
*.21432981

```

6.7.6 真正的 STATA 命令语句格式

执行上述命令将不会破坏任何原来的数据文件，但美中不足的是我们仍然没法运行更复杂的命令。想一想，STATA 自己的命令后面不仅仅可以用变量，还可以带上条件 if 或者范围 in。我们的命令也要这样子！ */

```

*=====begin=====

capture program drop gini
program gini
syntax varlist [if] [in] [,title(string)] //设定我们自己的命令格式
tempvar tinc tp m gini //设定 tinc tp m gini 四个变量为临时变量
marksample touse //生成一个 0/1 暂元，暂元名为 touse
preserve //将当前内存中数据暂封存，直到 restore 命令再复原
quietly { //大括号内的命令将在后台执行，前台无显示
keep if `touse' //根据 if 后输入的条件得到一个子数据
egen `tinc'=sum(`1') //生成总收入，将总收入数据暂存在临时变量'tinc'中
if "`2'"==" " {
local 2=1
} //如果没有人口变量，则默认为该变量为 1
egen `tp'=sum(`2')
gen `m'=`1'/`2'
sort `m'
gen `gini'=1-sum(`2'/`tp'*(2*sum(`1'/`tinc')-`1'/`tinc'))
}
display as result "`title'基尼系数为: " as error `gini'[_N] //显示基尼系数
restore

```

end

*=====end=====

use hb97,clear

gini cash hhsz

***基尼系数为: .35139242**

gini cash

***基尼系数为: .34467658**

gini cash hhsz if ds==3

***基尼系数为: .37227097**

gini cash hhsz if ds==3,title(山区食物消费)

***山区食物消费基尼系数为: .37227097**

gini cash hhsz if ds==1 in 1/200,title(平原前 200 户食物消费)

***平原前 200 户食物消费基尼系数为: .30506945**

*补充知识: quietly 与 noisily

di "hello, guy"

quietly di "hello, guy" //用了 quietly 后, 运行结果将不显示

capture program drop bb

program bb

quietly {

noisily di "hello, guy"

}

end

bb

//注意到显示结果: 没有任何显示

quietly bb

//注意到显示结果: hello, guy

7 流程语句

7.1 循环语句:while

任务 7.1: 用循环语句编写程序, 依次列出 1, 2, 3, 4, 5

程序示例:

```

=====begin=====
capture drop count5
program count5
local i=1          //先将宏名为 i 的宏值设为 1
while `i'<=5 {     //判断如果宏值 `i' 不大于 5, 就执行 {} 中的命令, 否则跳开
display `i'        //要干的活是显示宏值 `i'
local i=`i'+1      //重新设定宏名为 i 的宏值, 令其等于 `i'+1
}
end
=====end=====

count5
1
2
3
4
5

```

看看这些结果, `i' 最初设为 1, 因为 1 小于 5, 因此显示 1; 显示完毕紧随其后的命令, 将 i 重新设定 (即加上 1), 重新设定后 `i' 现在等于 2, 由于 2 仍然小于 5, 于是显示 2, 并再次重新设定 i,..., 直到 `i' 与 6 等价, 不再小于等于 5, 跳出循环 {}, 执行 end.

补充知识：数列的表示方法

```

2      just one number
1 2 3      three numbers
3 2 1      three numbers in reversed order
.5 1 1.5    three different numbers
1 3 -2.17 5.12    four numbers in jumbled order
1/3      three numbers: 1, 2, 3
3/1      the same three numbers in reverse order
5/8      four numbers: 5, 6, 7, 8
-8/-5    four numbers: -8, -7, -6, -5
-5/-8    four numbers: -5, -6, -7, -8
-1/2     four numbers: -1, 0, 1, 2
1 2 to 4  four numbers: 1, 2, 3, 4
4 3 to 1  four numbers: 4, 3, 2, 1
10 15 to 30    five numbers: 10, 15, 20, 25, 30
1 2:4        same as 1 2 to 4
4 3:1        same as 4 3 to 1
10 15:30     same as 10 15 to 30
1(1)3        three numbers: 1, 2, 3
1(2)9        five numbers: 1, 3, 5, 7, 9
1(2)10       the same five numbers: 1, 3, 5, 7, 9
9(-2)1       five numbers: 9, 7, 5, 3, and 1
-1(.5)2.5    the numbers: -1, -.5, 0, .5, 1, 1.5, 2, 2.5
1[1]3        same as 1(1)3
1[2]9        same as 1(2)9
1[2]10       same as 1(2)10
9[-2]1       same as 9(-2)1
-1[.5]2.5    same as -1(.5)2.5
1 2 3/5 8(2)12    eight numbers: 1, 2, 3, 4, 5, 8, 10, 12
1,2,3/5,8(2)12    the same eight numbers
1 2 3/5 8 10 to 12    the same eight numbers
1,2,3/5,8,10 to 12    the same eight numbers
1 2 3/5 8 10:12    the same eight numbers

```

7.3 循环语句:forvalues

程序示例：

```

*=====begin=====
forvalues i=1/5 {
  display `i'           //和上一个命令完全等价，只是写法更简洁
}
*=====end=====

```

程序示例 3:

```

=====begin=====
forvalues i=4 (-0.2) 0 { //起始值可大于终值，但步长应为负，步长可为小数
display `i'
}
=====end=====

```

用 **forvalues** 做循环时，其命令格式为

```

forvalues lname = range {
    commands referring to `lname'
}

```

其中 range 为

#1(#d)#2	从#1 开始，到#2 结束，步长为 #d
#1/#2	从#1 开始，到#2 结束，步长为 1
#1 #t to #2	从#1 开始，到#2 结束，步长为 (#t - #1)
#1 #t: #2	从#1 开始，到#2 结束，步长为 (#t - #1)

上述#可以为任何数：负数，小数。

任务 7.2: 用循环语句编写程序，计算 $1+2+3+\dots+100$

程序示例:

```

=====begin=====
scalar s=0 //scalar 命令生一个标量 s
forvalue i=1/100 {
scalar s=s+`i' //该标量 s 不断被替换，每一次都被加一次
}
scalar list s
=====end=====

```

7.3 循环语句:foreach

任务 5.3 (按变量循环): 打开数据文件 wage1, 对 nonwhite, female, married, numdep, smsa, northcen, south, west, construc, ndurman, trcommptu, trade, services, profserv, profocc, clerocc, servocc 这些变量分别求频数分布。(提示频数分布命令为 tab varlist)

```

=====begin=====
use http://fmwww.bc.edu/ec-p/data/wooldridge/ wage1, clear
tab nonwhite
tab female
tab married
tab numdep
tab smsa
tab northcen
tab south

```

```

tab west
tab construc
tab ndurman
tab trcommpu
tab trade
tab services
tab profserv
tab profocc
tab clerocc
tab servocc

```

```

=====end=====

```

对凡事讲究效率的现代人来说，这也够烦的，可是 `while` 还是 `forvalues` 都无能为力，因为我们所想要的是，对每个变量进行循环。现在是 `foreach` 一显身手的时候了。上面老长的命令，只需要下面几行字。

程序示例：

```

=====begin=====
use wage1, clear
foreach v of varlist nonwhite-servocc {
    tab `v'
}
=====end=====

```

用 `foreach` 做循环时，其命令格式为

```

foreach lname {in|of listtype} list {
    commands referring to `lname'
}

```

例如：

```

foreach lname in any_list {
    foreach lname of local lmacname {
    foreach lname of global gmacname {
    foreach lname of varlist varlist {
    foreach lname of newlist newvarlist {
    foreach lname of numlist numlist {

```

可以是数值、变量、文件等。

任务 7.4：将 `student.dta`，`economy.dta`，`math.dta` 纵向拼接起来

```

=====begin=====
use student, clear
foreach file in economy math {
    append using “file”
}

```

*或者

```

local flist economy math           //先将文件名赋于宏 flist
foreach file in `flist' {
    append using "file"
}

```

任务 7.5 (按项循环): 逐行显示粮食 (rice wheat flax maize) 和货币 (Dollar Lira Pound RMB)

```

=====begin=====
local grains "rice wheat flax maize"
foreach x of local grains {
    di "x"
}

global money "Dollar Lira Pound RMB"
foreach y of global money {
    di "y"
}
=====end=====

```

任务 7.6 (生成新变量): 生成五个新变量 b1, b2, b3, b4, b5, 每个变量都是均匀分布随机数

```

=====begin=====
clear
set obs 10
foreach v of newlist b1-b5 {
    gen `v'=uniform()
}
=====end=====

```

任务 7.7 (按数值循环): 逐行显示 1 2 3 4 8 105

```

=====begin=====
foreach num of numlist 1/4 8 105 {
    di `num'
}
=====end=====

```

这一用法与 forvalues 较相似, 但不要求数据完全是等步长连续, 这在很多情形下会十分方便, 比如在我们合并练习 2 中的数据文件时。

7.4 嵌套循环

任务 7.8: 生成 5 个变量, 10 个观察值, 使得第 i 个变量的第 j 个观察值等于 $i+j$

```

=====begin=====
clear
set obs 10
forvalues i=1/5 {           //i 的初始赋值为 1, 依次执行 2, 3, 4, 5 然后退出

```

```

    gen v`i'=. //将生成一个新变量 v1, 完成 v1 的赋值后将生成 v2, ...
    forvalues j=2(2)8 { // j 的初值为 2, 然后为 4, 6, 8, 然后退出内层循环
        replace v`i'=`i'+`j' in `j' //在第 2 行替换 v1=1+2=3, ...
    }
}

```

```

=====end=====

```

	v1	v2	v3	v4	v5
1.
2.	3	4	5	6	7
3.
4.	5	6	7	8	9
5.
6.	7	8	9	10	11
7.
8.	9	10	11	12	13
9.
10.

对照输出结果理解循环是如何执行的。

7.5 条件语句

7.5.1 简单的条件语句

如果是国产车，则价格降 100 元，如果是进口车，则价格提升 200 元。

```
sysuse auto, clear
```

```
gen p1=price-100 if foreign==0
```

```
replace p1=price+100 if foreign==1
```

```
gen p2=cond(foreign==0,(price-100),(price+100))
```

前两行与该行等价，到当条件满足时，执行紧跟条件后面的内容，否则执行第二个内容。

```
list p*forei
```

任务 7.9：判断奇偶数。

```

=====begin=====

```

```
capture program drop odd
```

```
program odd
```

```
args num
```

```
if int(`num'/2)!=(`num'-1)/2 { //如 (8-1)/2 取整为 3, 不等, 跳至 else
```

```
    display "num is NOT an odd number"
```

```
}
```

```
else {
```

```
    display "num IS an odd number"
```



```

}
end
*=====begin=====
    odd 1
    *num IS an odd number

    odd 100
    *num is NOT an odd number

```

任务 7.10: 检验数据文件中是否有某个变量。

```

*=====begin=====
capture program drop check
program check
capture confirm variable `l'
if _rc!=0 {
    display "`l' not found"
    exit
}
display "the variable `l' exists."
end
*=====end=====

use hb97,clear
check inc
the variable inc exists.
check exp
exp not found

```

7.5.2 循环语句套条件语句

任务 7.11: 求 6-500 之间能同时被 2, 3, 5 整除的数

```

*=====begin=====
forvalues x=6/200 {
    if mod(x',2)==0 & mod(x',3)==0 & mod(x',5)==0 { //mod() 为同余函数
        di "the ALL common multiple of 2,3 ,and 5 is  `x'"
    }
}
*=====end=====

the ALL common multiple of 2,3 ,and 5 is  30
the ALL common multiple of 2,3 ,and 5 is  60
the ALL common multiple of 2,3 ,and 5 is  90
the ALL common multiple of 2,3 ,and 5 is 120
the ALL common multiple of 2,3 ,and 5 is 150
the ALL common multiple of 2,3 ,and 5 is 180

```

任务 7.12: 求 6-500 之间能同时被 2, 3, 5 整除的最小的数

```

=====begin=====
forvalues x=6/500 {
    if mod(`x',2)==0 & mod(`x',3)==0 & mod(`x',5)==0 {
        di "the LEAST common multiple of 2,3 ,and 5 is  `x' "
        continue, break
    }
}
=====end=====
the LEAST common multiple of 2,3 ,and 5 is  30

```

7.6 复习和练习

- (1) 打开自己在为上机练习二课后习题而编写的程序，将其中可以用循环语句改写的命令重新写一遍并执行之。
- (2) 用自己编写的命令，利用 hb97 中的数据，快速计算出每个县的 FGT 指数。

8 矩阵

STATA9.0 版本的矩阵组件有了根本的革新，出现了与其他专门的矩阵运算软件功能一样强大的新组件 MATA, 用 MATA 可以完成除绘图以外的所有 STATA 数据处理和分析功能。这里介绍的仍然是传统的 STATA 矩阵命令，如果对 MATA 有兴趣，请参看笔者即将编写的另一本讲义（《MATA 讲义》）。

8.1 生成矩阵

8.1.1 输入新矩阵

可以直接录入一个新的矩阵

```
matrix A=(1,0,1\2,1,0\3,2,-5) //录入矩阵 A
```

```
matrix list A //显示录入的矩阵 A
```

```
matlist A //显示录入的矩阵 A,该命令只能在 stata9 以上版本使用
```

```
matrix B=(1+1,2*3/4\5/2,3^2)
```

```
matrix list B
```

```
matrix C=(2,3,4)
```

```
matrix E=(1\2\3)
```

```
matrix F=(4)
```

```
matrix rownames A= sex edu marriage
```

```
matrix colnames A=obs1 obs2 obs3
```

```
matrix list
```

8.1.2 生成特定格式矩阵

生成一个 5 行 3 列的矩阵 A，矩阵中的元素均为 0

```
matrix A=J(5,3,0)
```

生成一个 6 阶单位阵 I

```
matrix I=I(6)
```

生成一个 3 行 5 列的随机矩阵 R，每个随机元素均服从 (0, 1) 均匀分布

```
matrix R=matuniform(3,5)
```

```
matrix d=(1,4,9)
```

```
matrix D=diag(d)
```

//以d中元素为对角元素生成对角矩阵D

```
matrix list D
```

8.1.3 数据与矩阵之间的相互转化

将矩阵 R 转换为数据文件

```

mkmat varlist [if exp] [in range] [matrix(matname) nomissing ]
svmat [type] A [names(col|eqcol|matcol|string) ]
matname A namelist [rows(range) columns(range) explicit ]

```

```

*=====begin=====

```

```

sysuse auto, clear

```

```

*将 foreign weight displ 这三个变量的前 5 个观察数据转换为矩阵 X

```

```

mkmat foreign weight displ in 1/5, matrix(X)

```

```

matrix list X

```

```

*将带条件的数据转换为矩阵 Y

```

```

mkmat foreign weight displ if fore==1, matrix(Y)

```

```

matrix list Y

```

```

matrix A= (1,0,1\2,1,0\3,2,-5) //录入矩阵 A

```

```

svmat A //将矩阵 A 中的数据转化为数据，变量为 A1, A2, A3

```

```

list

```

```

svmat A, name(ccb) //将矩阵 A 中的数据转化为新的变量 ccb1, ccb2, ccb3

```

8.2 矩阵四则运算

表 5-1：矩阵运算一览表

运算	STATA 命令	说明
$C = A \pm B$	<code>mat c = a ± b</code>	$\dim(a)=\dim(b)$
$C = A * B$	<code>mat c = a*b</code>	$\text{cols}(a)=\text{rows}(b')$
$C = B^{-1}$	<code>mat c = inv(b)</code>	b 为方阵
$C = A'$	<code>mat c = a'</code>	转置运算
$C = A \otimes B$	<code>mat c = a # b</code>	直乘运算
$C = A/k$	<code>mat c = a/ k</code>	k 为常数

以上表格引用自arlion 《Estimation with STATA—Matrix》

8.2.1 矩阵扩展

```

matrix A= (1,0,1\2,1,0\3,2,-5)

```

```

matrix B=(4\3\7)

```

```

matrix C=(A,B)

```

```

matrix list C

```

```

matrix D= (10,9,25)

```

```

matrix E= (A\D)

```

```

matrix list E

```

8.2.2 矩阵加、减、数乘、除法

```

mat e=(3,5\2,0)
mat t=(1,3\2,1)

mat r1=e+t           //加
mat list r1

mat r2=e-t           //减
mat list r2

mat e2=2*e           //数乘（每个元素均乘标量）
mat list e2

mat e3=e/10          //除法（每个元素均除以一个标量）
mat list e3

```

8.2.3 矩阵乘法、直乘及 Kronecker 乘积运算

```

mat w=e*t           //乘
mat list w

```

*矩阵乘法不满足交换律，一般来说 $a*b \neq b*a$

```

mat a=(2,3\1,-2\3,1)
mat b=(1,-2,-3\2,-1,0)
mat ab=a*b
mat list ab
mat ba=b*a
mat list ba

```

/*两个矩阵对应元素直接相乘，要求两个矩阵结构相同。*/

```

mat A=(1,2\3,4\5,6)
mat B=(7,8\9,0\1,9)
matrix H=hadamard(A,B)
mat list H

```

/*矩阵的 Kronecker 乘积运算定义如下：

$$\mathbf{A} \otimes \mathbf{B} = \begin{bmatrix} a_{11}\mathbf{B} & a_{12}\mathbf{B} & \cdots & a_{1K}\mathbf{B} \\ a_{21}\mathbf{B} & a_{22}\mathbf{B} & \cdots & a_{2K}\mathbf{B} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1}\mathbf{B} & a_{n2}\mathbf{B} & \cdots & a_{nK}\mathbf{B} \end{bmatrix}$$

Kronecker 乘积运算对 A 和 B 没有任何要求。*/

```
mat ab2=a#b
```

```
mat list ab2
```

```
mat ba2=b#a
```

```
mat list ba2
```

8.3 矩阵函数

函数名称	说明
cholesky(B)	裘氏分解，返回 P，使 $PP' = B^{-1}$ ，B 为对称、正定矩阵；
corr(B)	相关转换 ⁵
diag(V)	若 V 为 $1 \times n$ 或 $n \times 1$ ，则返回对角元素为 V 的 $n \times n$ 矩阵；
get(name)	return system matrix (see help get())
hadamard(A,B)	返回矩阵 C，满足 $C[i, j] = A[i, j] * B[i, j]$ ；
I(n)	返回 $n \times n$ 单位矩阵；
inv(B)	返回方阵 B 的逆矩阵 B^{-1} ；
J(r,c,z)	返回 $r \times c$ 矩阵，所有元素均为常数 z；
matuniform(r,c)	
nullmat(A)	对于 <code>mat c=(nullmat(a),b)</code> 。若 a 不存在， <code>c=b</code> ；否则， <code>c=(a,b)</code> ；
sweep(A,z)	sweep of square matrix; return A with zth row/column swept
syminv(B)	inverse of symmetric matrix; if B is not positive definite,
vec(A)	将矩阵 A 向量化后返回 B，即所有列依次堆积，若 $A_{m \times n}$ ，则 $B_{mn \times 1}$
vecdiag(B)	返回一个行向量，包含方阵 B 的对角元素；

函数名称	返回值说明
colnumb(A,s)	返回矩阵 A 中第一个名称为字符串 s 的列所在的列数；
colsof(B)	返回矩阵 B 的列数；
det(B)	返回矩阵 B 的行列式；
diag0cnt(B)	返回矩阵 B 中对角线为零的元素的个数；
el(A,i,j)	返回矩阵 A 中第 [i,j] 个元素，等价于 A[i,j]；
issym(A)	判断矩阵 A 是否为对称矩阵，是返 1，否返 0；
matmissing(A)	判断矩阵 A 中是否含有缺漏值，是返 1，否返 0；
mreldif(B,C)	返回矩阵 B 和 C 的相对差异； ⁴
rownumb(A,s)	返回矩阵 A 中第一个名称为字符串 s 的行所在的行数；
rowsof(B)	返回矩阵 B 的行数；
trace(B)	返回方阵 B 的秩；

以上表格引用自arlion 《Estimation with STATA—Matrix》

8.3.1 转置、求迹

```
mat b=(1,-2,-3\2,-1,0)
mat list b
mat b2=b'           //b2 为 b 的转置矩阵
mat list b2
```

/*迹的表达式为，它只能对方阵求，其结果是一个数

$$\text{tr}(\mathbf{A}) = a_{11} + a_{22} + \cdots + a_{nn} \quad */$$

```
matrix A=(1,0,1\2,1,0\3,2,-5)
scalar a=trace(A)      //求 A 的主对角线上的元素之和并赋予该值给 b
scalar list
```

8.3.2 求逆、求行列式的值

```
matrix B=inv(A)        //B 为 A 的逆矩阵
mat list B
```

/*任务：解线性方程 $Ax=b$ ，其中：

```
A=(1,-1,1,-2\2,0,-1,4\3,2,1,0\1,2,-1,2)
```

```
b=(2\4\1\4) */
```

```
mat AI=(1,-1,1,-2\2,0,-1,4\3,2,1,0\1,2,-1,2)
mat b=(2\4\1\4)
```

```

matrix B1=inv(A1)           //B 为 A 的逆矩阵
mat X=B1*b
mat list X

mat a=(1,0,1\2,1,0\3,2,-5)
scalar aa=det(a)           //det()计算行列式的值
mat d=(1,-1,1,-2,2\2,0,-1,4,4\3,2,1,0,-1\1,2,-1,2,-4)
mat dd=det(d)
scalar list

```

8.3.3 返回或者修改矩阵中某个元素的值

```
el(A,i,j)=A[i,j]
```

```

mat d=(1,-1,1,-2,2\2,100,-1,4,4\3,2,1,0,-1\1,2,-1,2,-4)
scalar dd=d[2,2]
scalar ddd=el(d,2,2)
scalar list

```

```

mat d[1,3]=1000
mat list d

```

8.3.4 Nullmat 命令

该命令在编程时特别有用，有点类似于数据管理中的 `merg` 命令

```

mat A=I(3)
mat B=J(3,3,9)
mat list C    //矩阵C不存在

mat C=(nullmat(C),A) //当C矩阵不存在时，将得到C=A
mat list C      //C=A

```

```

mat C=(nullmat(B),A) //C变成由B和A两个矩阵横向对接
mat list C           //C= (B, A)

```

```

mat A=A/10
mat C=(nullmat(C),A) //C在原来的基础上对接了A变成新的矩阵
mat list C           //C= (C, A)

```

8.3.5 其他矩阵函数

```

matrix A= (1,0,1\2,1,0\3,2,-5)
matrix e=vec(A)           //将矩阵A转化为一维列矩阵e
mat list e

matrix d=vecdiag(A)       //返回矩阵A的对角元素为一行向量d
matrix list d

```



```

mat J=J(10,2,)
scalar a=rowsof(J)           //求矩阵的行数
scalar b=colsof(J)           //求矩阵的列数
scalar list

```

8.4 随机向量与矩阵代数（选学内容）

该部分的目的是更好地理解随机向量的现实运行机理。

设 (X, Y) 的概率分布由右表给出。（数理统计书本73页的例题）

X \ Y	-1	0	2
0	0.1	0.2	0
1	0.3	0.05	0.1
2	0.15	0	0.1

(1) 求 $p\{X \neq 0, Y=0\}$; $P\{X \leq 0, Y \leq 0\}$; $P\{XY=0\}$; $P\{X=Y\}$; $P\{|X|=|Y|\}$

(2) 求 $Y=0$ 时, X 的条件概率分布; $X=0$ 时, Y 的条件概率分布

(3) 判断 X 与 Y 是否相互独立

(4) 求 $\zeta=X+Y$ 的概率分布及期望; $\xi=XY$ 的概率分布及期望

(5) 求 EXY

(6) 求 $\text{cov}(x, y)$

(7) 求 $E(X|Y=0)$

(8) 设 $Z' = (X, Y)$, $A = (1, -1/2, 1)$, $W=AZ$, 验证: $E(W) = AE(Z)$

(9) 求协方差阵 $\text{cov}(Z)$

(10) 证明协方差阵为半正定对称阵

(11) 证明 $\text{cov}(W) = A\text{cov}(Z)A'$

(12) 设 $B = (1, -2/0, -1)$ 证明: $\text{cov}(AX, BY) = A\text{cov}(X, Y)B'$

(13) 证明: $E(X'AX) = u'Au + \text{tr}(A\Sigma)$; 其中 $u=E(X)$, $\Sigma=\text{cov}(X)$

=====begin=====

**随机向量 $X = (x_1, x_2)'$, 且 x_1, x_2 的取值范围为

```
mat x1=(0,1,2)
```

```
mat x2=(-1,0,2)
```

**若随机抽取样本, 共有9种取法, 如取得 $(x_1=1, x_2=2)$,

```
mat X=(0,-1\0,0\0,2\1,-1\1,0\1,2\2,-1\2,0\2,2)
```

**相应于 $(0,-1), (0,0), (0,2); (1,-1) \dots (2, 2)$ 等样本的概率分布为

```
mat p=(.1,.2,0,.3,.05,.1,.15,0,.1)
```

**生成9组随机样本

**求均值向量

```
mat EX=X'*p'
```

**中心化

```
mat dX=X'-EX#J(1,9,1)
```

**求协方差阵

```
mat CovX=dX*(hadamard(dX,J(2,1,1)*p))'
. mat list CovX
```

**给定非随机阵A

```
mat A=(1,-1\2,1)
```

**得到X的线性组合而成的新随机变量Y

```
mat Y=A*X
```

**Y的中心化

```
mat dY=Y-Y*p'#J(1,9,1)
mat list dY
```

**Y的协方差阵：直接计算法

```
mat CovY=dY*(hadamard(dY,J(2,1,1)*p))'
```

```
mat list CovY
```

**利用定理： $\text{Cov}(Y)=A*\text{Cov}(X)A'$ 间接计算其协方差阵

```
mat CovY2=A*CovX*A'
mat list CovY2
```

**随机向量X的二次型

```
mat XAX=vecdiag(X'*A*X)
```

**直接计算随机向量二次型的期望

```
mat EXAX=XAX*p'
```

**利用公式 $E(X'AX)=u' Au+\text{tr}(A\text{Cov}X)$ 来计算二次型的期望

```
mat EXAX2=EX'*A*EX+trace(A*CovX)
mat list EXAX2
```

*=====end=====

第（4）题，求 $\zeta=X+Y$ 的概率分布及期望，理论计算结果为

ζ	-1	0	1	2	3	4
p	0.1	0.5	0.2	0	0.1	0.1

期望 $E \zeta = -1*0.1 + 0*0.5 + 1*0.2 + 2*0 + 3*0.1 + 4*0.1 = 0.8$

模拟：

*=====begin=====

```
capture program drop rv
```

```
prog rv
```

```

local s=0
matrix a=(0.1,0.2,0\0.3,0.05,0.1\0.15,0,0.1) //定义 X 与 Y 的联合概率分布
matrix x=(0,1,2) //定义随机变量 X
matrix y=(-1,0,2) //定义随机变量 Y
forvalues i=1/3 {
    forvalues j=1/3 {
        if uniform()>`s'& uniform()<=`s'+a[`i'`,`j'] {
            scalar z=x[1,`i']+y[1,`j'] //定义随机变量 Z=X+Y
        }
        local s=`s'+a[`i'`,`j']
    }
}
end

simulate z ,reps(10000) nodots:rv //10000 次模拟
tab _simu //求分布
sum //求期望
*=====end=====

```

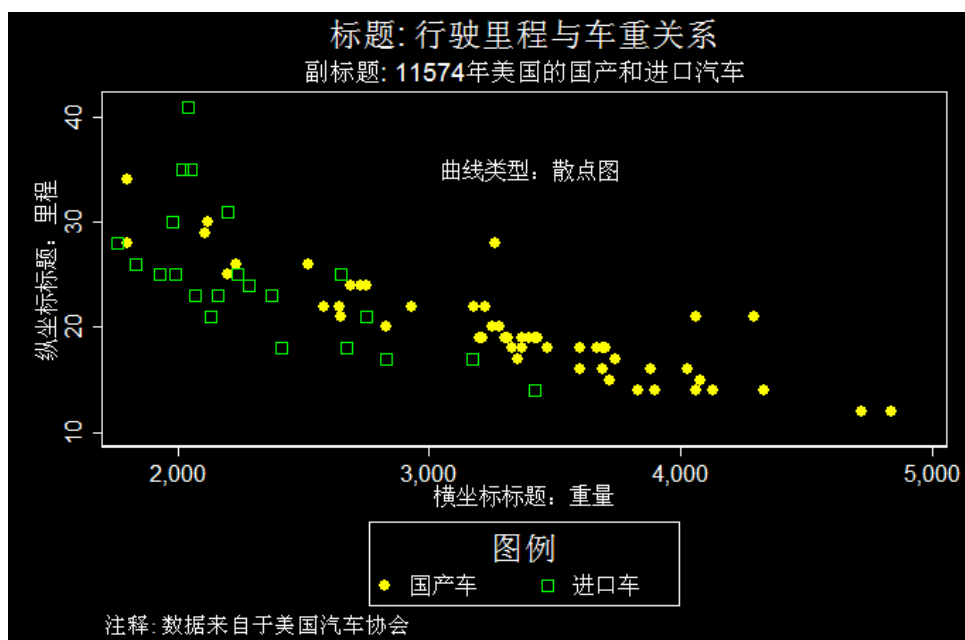
9 绘图

STATA10.0 版本的绘图功能有了新的提升，主要是提供了窗口化的图形处理能力，使得绘图可以像在 EXCEL 中一样，用鼠标完成操作。但是这些操作背后的命令基础仍然是本讲即将介绍的内容。

注：本讲中许多例子来自于“a visual guide to stata graphics”一书。

9.1 绘图命令

一个完整的图应包括以下要素：曲线（点/线/面）、标题与副标题、图例、脚注、插图、坐标轴。



以下命令显示出上图

```
*=====begin=====
sysuse auto , clear
twoway (scatter mpg weight if foreign==0)          ///
      (scatter mpg weight if foreign==1 , msymbol(Sh)) /*曲线选项，点的类型*/  ///
      ,
      title(标题: 行驶里程与车重关系)             /*图选项: 标题*/      ///
      subtitle(副标题: 11574年美国的国产和进口汽车)      ///
      ytitle(纵坐标标题: 里程)                     ///
      xtitle(横坐标标题: 重量)                     ///
      note(注释: 数据来自于美国汽车协会)          ///
      text(35 3400 "曲线类型: 散点图")             ///
      legend(title(图例) label(1 国产车) label(2 进口车))  ///
      scheme(s1rcolor)
*=====end=====
```

9.1.1 命令结构

graph-command (plot-command, plot-options) (plot-command , plot-options) , graph-options
 或者
graph-command plot-command,plot-options || plot-command , plot-options || , graph-options

graph-command定义图的类型，plot-command 定义曲线类型,同一个图中如果有多条曲线可以用括号分开，也可以用“||”分开，曲线有其自身的选项，而整个图也有其选项。例如**twoway**为graph-command中的命令之一，而**scatter**为plot-command中的命令之一。

曲线选项和图选项，例如

```
twoway (scatter mpg weight) , title("美国汽车")      //图选项：标题
twoway (scatter mpg weight , msymbol(Oh))            //曲线选项，点的类型
twoway (scatter mpg weight , msymbol(Oh)) , title("美国汽车") //同时用图与曲线选项
```

命令可以简写，如下列命令等价

```
sysuse auto, clear
graph twoway scatter mpg weight
      twoway scatter mpg weight      //graph可以省略
      scatter mpg weight             //twoway graph都可以省略
gr tw sc mpg weight
tw sc mpg weight
sc mpg weight
```

9.1.2 曲线类型

STATA 提供各种曲线类型，包括点 (scatter)、线 (line)、面 (area)，直方图 (histogram)、条形图 (bar)、饼图 (pie)、函数曲线 (function) 以及**矩阵图 (matrix)** 等。对时间序列数据有以 ts 开头的一系列特殊命令，如 **tsline**。还有一类是对双变量的回归拟合图 (lfit、qfit、**lowess**) 等。可以用帮助命令查看。

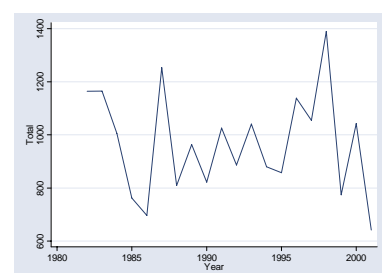
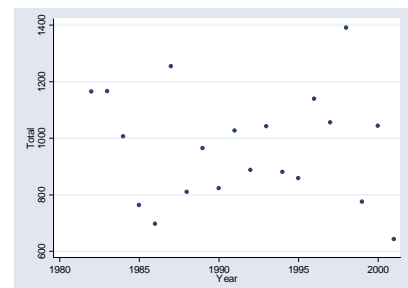
help graph

help twoway

(1) 曲线**从大的方面可以分为点线面**，在同一个图中可以容纳多条曲线，甚至是不同类型的曲线。**先来看单个曲线类型。**

任务 9.1: **数据集 rf.dta** 中有某县,其中变量 **rainfall** 为年降雨量, **year** 为年份, 画出该县各年降雨量散点图、折线图 and 面积图,

```
点
use rf, clear
keep if id==1
save rfsz, replace
scatter rainfall year
线
line rainfall year
面
tw area rainfall year
```



(2) 几种基本的曲线类型

任务 9.2: 使用美国 51 个州的人口 *popk*、居住成本 *propval100* 及普查分区 *division* 等变量绘制各种图，数据文件为 *allstates*。这些文件在网络上，首先要确保网络连接正常。

vguse allstates ,clear

直方图: histogram

twoway histogram popk

核估计图: kdensity

twoway kdensity popk

条形图: bar 和 hbar

graph bar popk, over(division)

graph hbar popk, over(division)

箱形图 box 和 hbox

graph hbox popk, over(division)

graph box popk, over(division)

饼图: pie

graph pie popk, over(region)

矩阵图 matrix

graph matrix propval100 rent700 popden

点阵图 dot

graph dot popk, over(division)

(3) 更多细分的曲线类型

任务 9.3: 使用股价数据 *spjanfeb2001* 绘制各种图，其中 *close* 为收盘价，*open* 为开盘价，*tradeday* 为交易日。

vguse spjanfeb2001, clear

twoway dropline close tradeday //

tw spike close tradeday //针式图

tw dot close tradeday //点线图

tw connected close tradeday, sort //点连线图

tw area close tradeday, sort //和线图类似，显示线以下的面积

tw bar close tradeday, sort /*对每个X绘出相应Y的条图，而graph bar是针对分类变量X的*/

twoway rarea high low tradeday, sort //显示最高与最低价，并填充两者之间的面积

tw rline high low tradeday, sort //显示最高与最低价，但不填充其间

tw rconnected high low tradeday, sort //显示最低与最高为两点，并连接起来

tw rscatter high low tradeday, sort //显示最低与最高为两点，不连接

tw rcap high low tradeday, sort //两端用小横线标示

tw rspike high low tradeday, sort //两端用点标示

tw rcapsym high low tradeday, sort

对时间序列数据，可以先设定时期，简化命令

tsset tradeday //X轴为时间，由tsset设定.

twoway tsline close, sort //tsline用于时间序列数据，收盘价

```
tw tsline high low, sort //最高价与最低价
```

9.1.3 标题项: title()

整个图标题: title()

图的副标题: subtitle()

为图加上标题和副标题

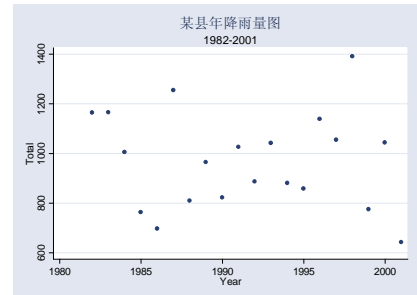
```
use rfsz, clear
```

```
scatter rainfall year ///
```

```
, ///
```

```
title(某县年降雨量图) ///
```

```
subtitle(1982-2001)
```



9.1.4 坐标轴格式

(1) 有坐标轴、有刻度格式 (默认)

```
scatter rainfall year
```

(2) 无坐标轴格式

可以不标y轴, 命令为`yscale(off)`; 也可不标出x轴`xscale(off)`, 还可以不显示图中的暗格。

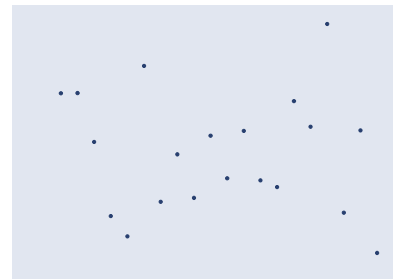
```
scatter rainfall year ///
```

```
, ///
```

```
yscale(off) ///
```

```
xscale(off) ///
```

```
plotregion(style(none))
```



(3) 无坐标轴, 有刻度格式

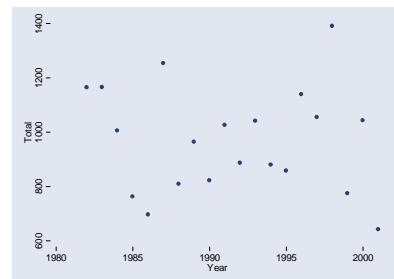
```
scatter rainfall year ///
```

```
, ///
```

```
plotregion(style(none)) ///
```

```
yscale(noline) ///
```

```
xscale(noline)
```



(4) 双坐标轴格式

同时绘出降雨量和单产的折线图时, 单产变量`yield`,

```
line rainfall yield year //该命令等价于
```

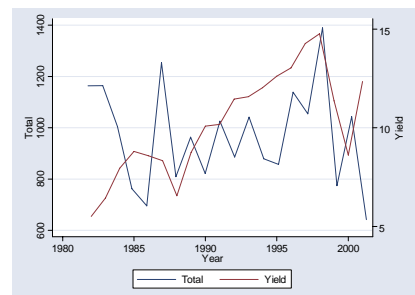
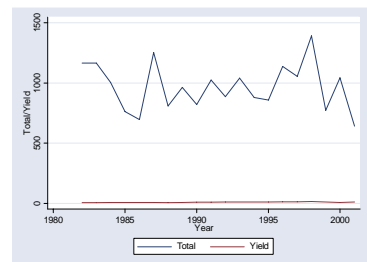
```
tw (line rainfall year ///
```

```
(line yield year)
```

我们发现单产几乎为一条直线, 实际上单产的波动是很大的, 之所以像一条直线, 是因为降雨量 (mm) 和单产 (T/ha.) 的单位不一致导致的。如果我们用双 Y 轴, 将单产的纵轴用右纵轴表示。

```
twoway (line rainfall year, yaxis(1)) ///
```

```
(line yield year, yaxis(2))
```



(5) 坐标轴标题

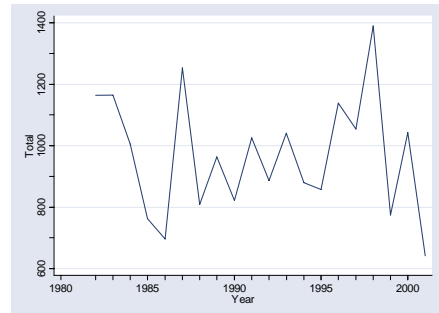
纵坐标标题: `ytitle()`

横坐标标题: `xtitle()`

```
use rfsz,clear
```

```
label var yield 单产
```

```
tw (line rain year, yaxis(1) ytitle(降雨量)) ///
   (line yield year, yaxis(2) xtitle(年份))
```



(5) 坐标轴刻度

左纵坐标刻度: `ytick()`

下横坐标刻度: `xtick()`

```
line rainfall year          ///
,                          ///
xtick(1982(1)2000)         ///
ytick(600(100)1400)
```

(6) 坐标轴刻度值

左纵坐标刻度及刻度值: `ylabel()`

下横坐标刻度及刻度值: `xlabel()`

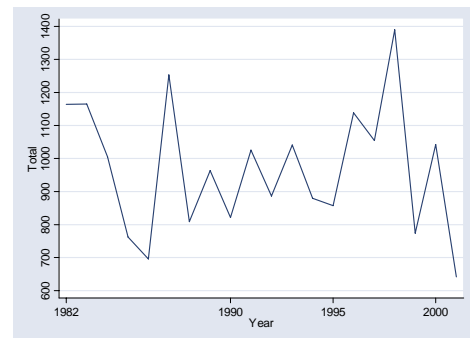
```
line rainfall year          ///
,                          ///
xlabel(1982 1990 1995 2000) ///
ylabel(600(100)1400)
```

更多的刻度及标注方法, 请查找帮助

`help axis label option`

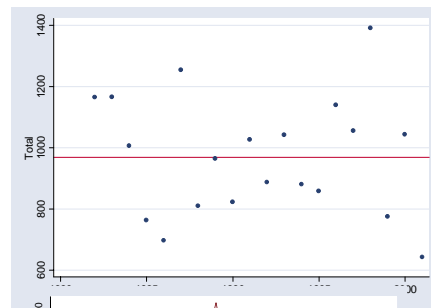
做一个图, 纵横比例为 3: 4

```
scatter mpg weight, ysize(3) xsize(4)
```

9.1.5 任意水平线与垂直线: `yline()`与 `xline()`

任务 9.4: 在降雨量散点图中画出平均雨量线

```
scatter rainfall year          ///
,                          ///
xlabel(1982 1990 1995 2000) ///
ylabel(600(100)1400)
```

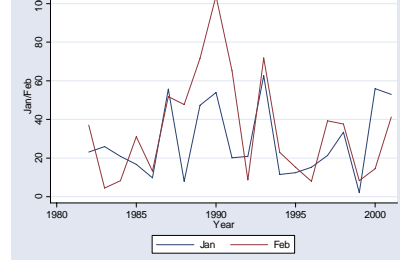


9.1.6 图例 legend ()

(1) 自动插入图例

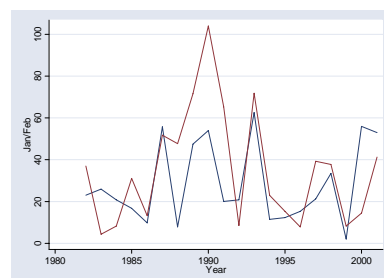
任务 9.5: 当有两条曲线时, STATA 会自动生成图例, 置于图的下方。如画出一月和二月份的降雨量图。

```
line jan feb year
```



(2) 关闭自动插入的图例

```
line jan feb year          ///
```

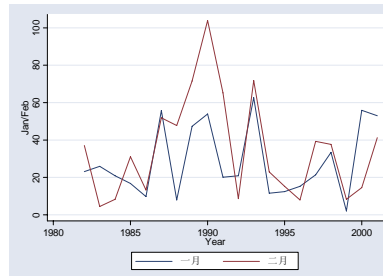



```
,
legend(off)
```

(3) 定制图例内容 legend(label())

将图例中的英文改为中文

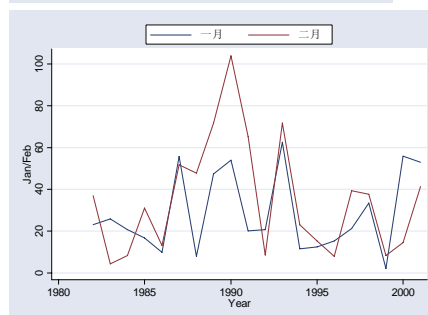
```
line jan feb year
,
legend(label(1 一月) label(2 二月))
也可以先将变量标识而后自动生成
label var jan 一月
label var feb 二月
line jan feb year
```



(4) 定制图例位置 legend(position())

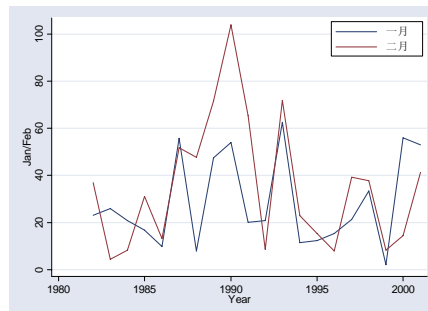
可将图例放在图的 12 个位置（对应于时钟的小时刻度）如正右边为 position(3) 右正方为 position(5)，正上方为 position(12)。

```
line jan feb year
,
legend(pos(12) label(1 一月) label(2 二月))
```



有时我们希望将图例放在图中，命令为 ring(0)

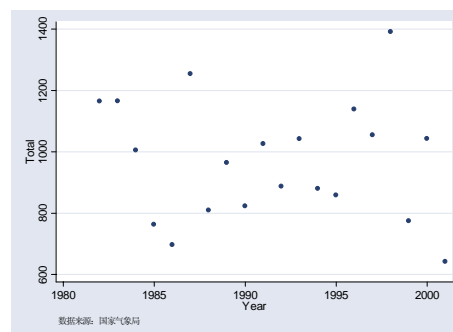
```
line jan feb year
,
legend(pos(1)
ring(0)
label(1 一月)
label(2 二月)
col(1)) //col(1) 要求图例按一列处理
```



9.1.7 脚注：note()

脚注主要用于标明数据的来源或者有关对整个图的说明，如

```
scatter rain year
,
note(数据来源：国家气象局)
```



9.2 几种常用的图

9.2.1 散点图与连线图

指定图形中散点的表示符号：msymbol()

圆圈	菱形	正方形	三角形	+号	X号	小点
----	----	-----	-----	----	----	----

大、实心	O	D	S	T	+	X	
大、空心	Oh	Dh	Sh	Th			
小、实心	o	d	s	t		x	p
小、空心	oh	dh	sh	th			

任务 9.6，绘制美国各州贫困发生率的散点图，用做人口数做权重。

```
vguse allstates, clear
```

```
scatter pov statefips [fw=pop], msymbol(oh)
```

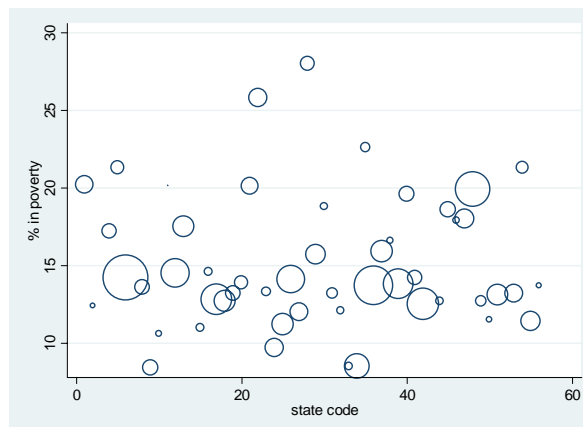
指定散点的颜色

```
scatter pov statefips [fw=pop], msymbol(o)
```

```
mfcolor(green) mlcolor(red)
```

任务：绘制出各州贫困发生率散点图，并标上州名

```
scatter pov statefips, mlabel(stateab)
```

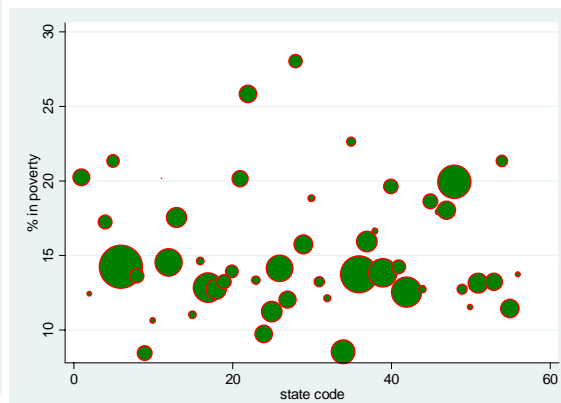


单独标出某个州

```
vguse allstates, clear
```

```
twoway (scatter ownhome borninstate if stateab=="DC", mlabel(stateab)) ///
```

```
(scatter ownhome borninstate), legend(off)
```



指定图形中散点的连接方式：connect()

. 不连接

l 用直线连接

L 按 X 在数据库中的顺序用直线连接

m 用直线连接中位值

s 用三次平滑曲线连接

J 以阶梯式直线连接

|| 连接垂直方向上的两个点

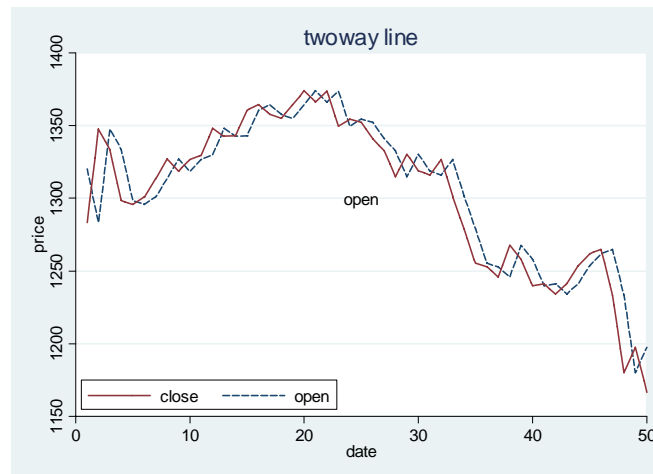
II 在顶及底部添加短横线

对每种线条类型，可单独定义线条样式，如 c(l)为实线相连，c(lf-)以虚线相连

l 实线

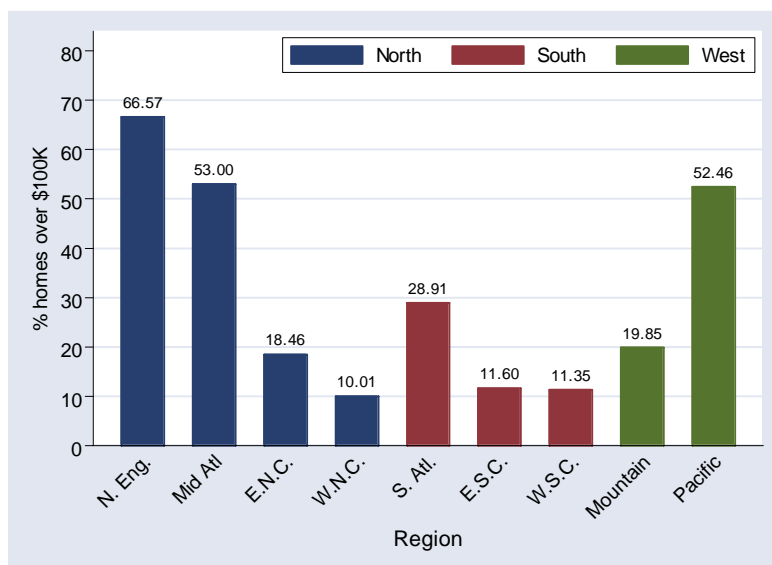
l[_] 长线段
 l[-] 中等长线段
 l[.] 短线段
 l[#] 空格

```
sysuse sp500,clear
gen n=_n
twoway ///
(line open close n in 1/50 , sort clpattern( -))
///
, ///
plotregion(margin(zero)) ///
title("twoway line") ///
xtitle("date") ///
ytitle("price") ///
legend(label(1 "open") label(2 "close"))
order(2 1) ///
ring(0) pos(8)) ///
text(1300 25 "open")
```



9.2.2 条形图

任务 9.7: 以下是美国三大区域(nsw)9 个普查区域(division)住房平均成本(propval100), 括号中为相应的变量名称, 请使用数据 allstates.dta 绘出如下的条形图

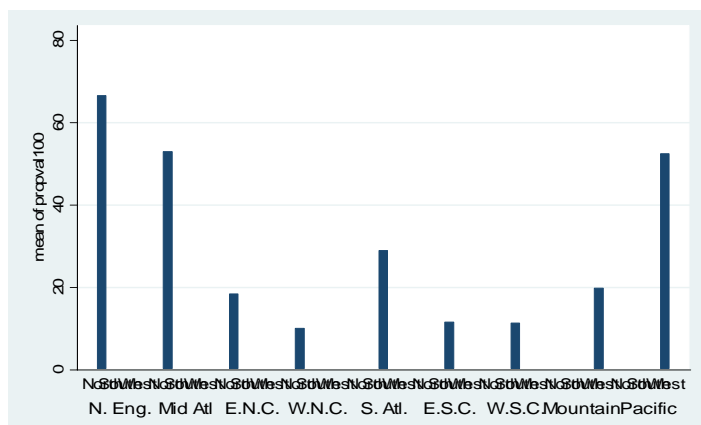


最原始的命令

```
vguse allstates, clear
gsort nsw division
list propval100 nsw division
```

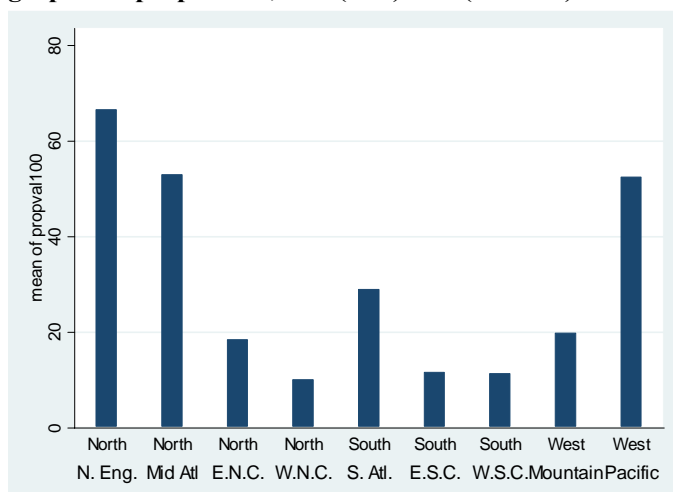
以上命令用于查看数据, 从数据中可以看出, 同一个区域中有若干次区域, 次区域中有若干被子调查的住房成本, 绘图时采用的为每个次区域的平均成本。

```
graph bar propval100, over(nsw) over(division)
```



上图非常难看，横坐标的标示糊成了一团，`nofill` 命令将分类变量中缺失值忽略掉（在前面将其做为一组来处理）

graph bar propval100, over(nsw) over(division) nofill



上面的图中，各个条形图之间间距较大，仍然不够美观。而下面的命令选项 **asyvars** 将使得第一个分组变量转换成相应的若干 y 变量。如 `ynorth ysouth ywest`

graph bar propval100, over(nsw)
over(division) nofill asyvars

上述命令等价于

use allstates, clear

gen yNorth=propval100 if nsw==1

gen ySouth=propval100 if nsw==2

gen yWest=propval100 if nsw==3

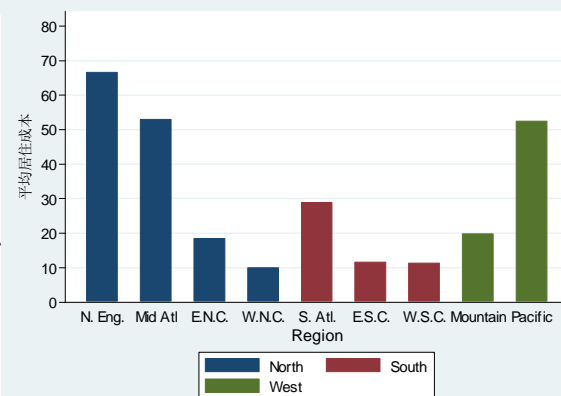
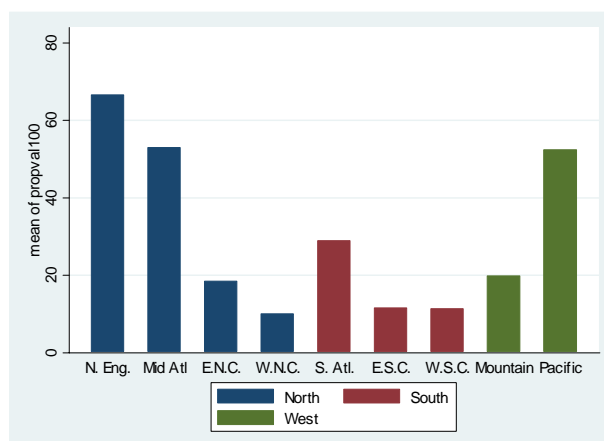
graph bar ///

yNorth ySouth yWest, ///

over(division) nofill

下面的命令对坐标轴的刻度、标识及标题进行调整

graph bar propval100, over(nsw)
over(division) nofill asyvars ///



```

ytitle(平均居住成本)           ///
ylabel(0(10)80)                ///
b1title(Region)

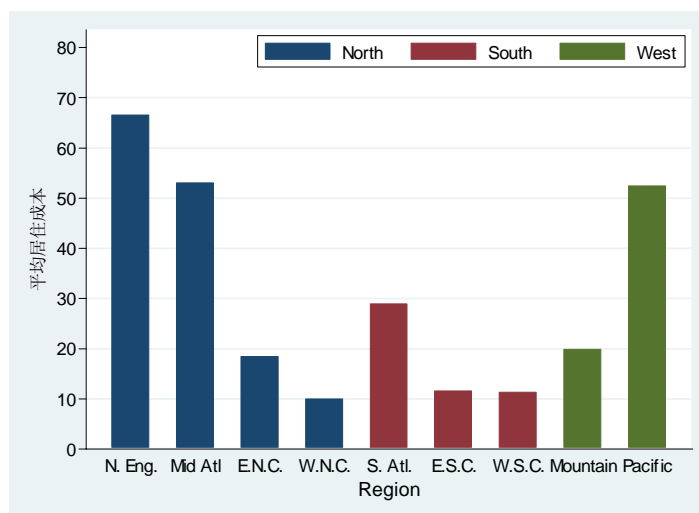
```

下面的命令美化图例

```

graph bar propval100, over(nsw) over(division) nofill asyvars ///
ytitle(平均居住成本) ylabel(0(10)80) b1title(Region)  ///
legend(rows(1) position(1) ring(0))

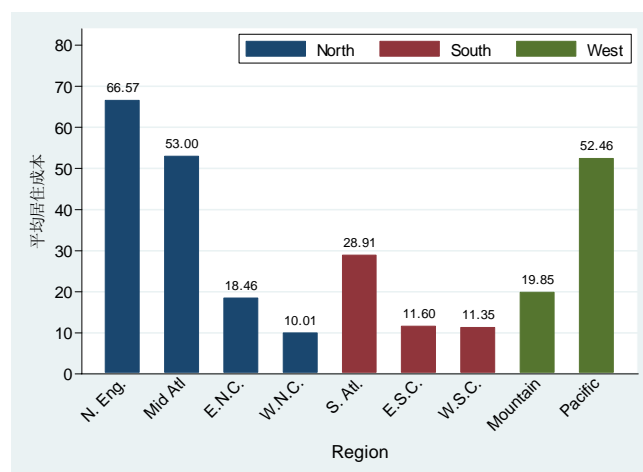
```



```

graph bar propval100, over(nsw)           ///
over(division, label(angle(45))) /*将分组变量标识倾斜45度*/      ///
nofill asyvars ytitle("平均居住成本") ylabel(0(10)80, angle(0)) b1title(Region)  ///
legend(rows(1) position(1) ring(0))           ///
blabel(bar, format(%4.2f)) //给每个条形图加上数据并规定数据格式

```



例:

```
clear
```

```
input str5 age m f
```

```
16-24 .9 .2
```

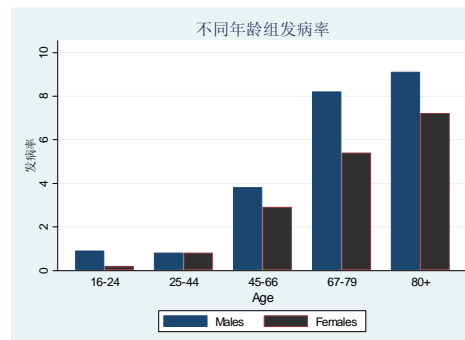
```
25-44 .8 .8
```

```
45-66 3.8 2.9
```

```

67-79 8.2 5.4
80+ 9.1 7.2
end //以上程序录入数据
graph bar m f /// //绘条形图命令 graph bar,
, ///
over(age) /// //按年龄绘制
title(不同年龄组发病率) ///
b1title("Age") /// //底部标题
ytitle(发病率) ///
legend( label(1 "Males") label(2 "Females") ) ///
bar(2 , bfcolor(gs3)) //第二类条形图的色彩方案

```



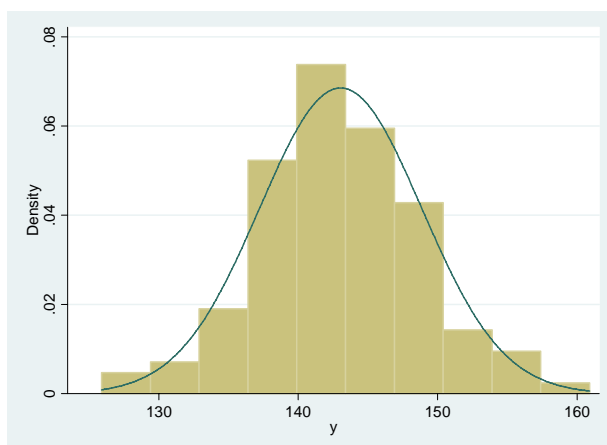
9.2.3 直方图

任务 9.8 绘制身高直方图，录入如下数据，将数据分为 10 组，统计落入每组的频数，然后绘制直方图。

```

*=====begin=====
clear
set memory 64m
input x1-x10
142.3 156.6 142.7 145.7 138.2 141.6 142.5 130.5 134.5 148.8
134.4 148.8 137.9 151.3 140.8 149.8 145.2 141.8 146.8 135.1
150.3 133.1 142.7 143.9 151.1 144.0 145.4 146.2 143.3 156.3
141.9 140.7 141.2 141.5 148.8 140.1 150.6 139.5 146.4 143.8
143.5 139.2 144.7 139.3 141.9 147.8 140.5 138.9 134.7 147.3
138.1 140.2 137.4 145.1 145.8 147.9 150.8 144.5 137.1 147.1
142.9 134.9 143.6 142.3 125.9 132.7 152.9 147.9 141.8 141.4
140.9 141.4 160.9 154.2 137.9 139.9 149.7 147.5 136.9 148.1
134.7 138.5 138.9 137.7 138.5 139.6 143.5 142.9 129.4 142.5
141.2 148.9 154.0 147.7 152.3 146.6 132.1 145.9 146.7 144.0
135.5 144.4 143.4 137.4 143.6 150.0 143.3 146.5 149.0 142.1
140.2 145.4 142.4 148.9 146.7 139.2 139.6 142.4 138.7 139.9
end
stack x1-x10,into(y) clear
ren _sta n
forvalues i=1/10 {
  replace n=`i' if y>124+`i'*4&y<124+4*(`i'+1)
}
hist y,norm
*=====end=====

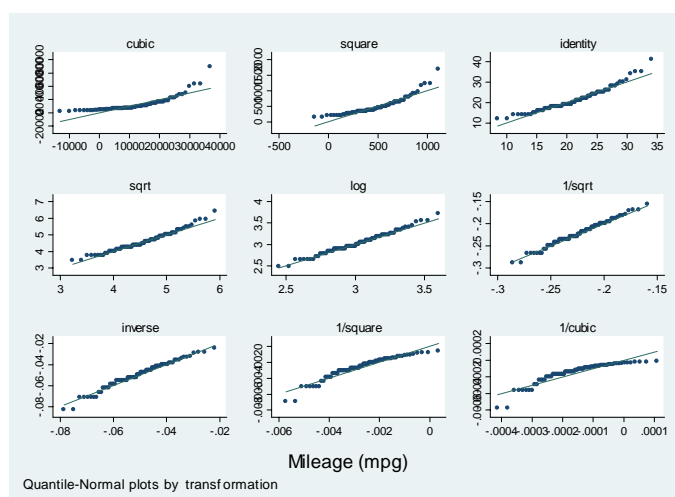
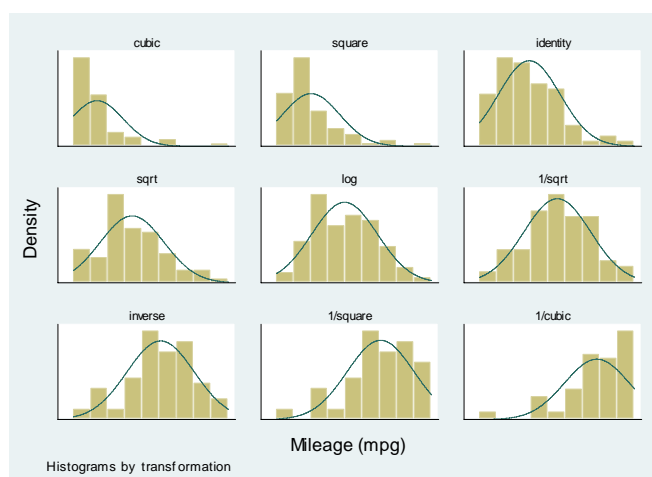
```



sysuse auto, clear

ladder mpg

gladder mpg



9.2.4 箱图

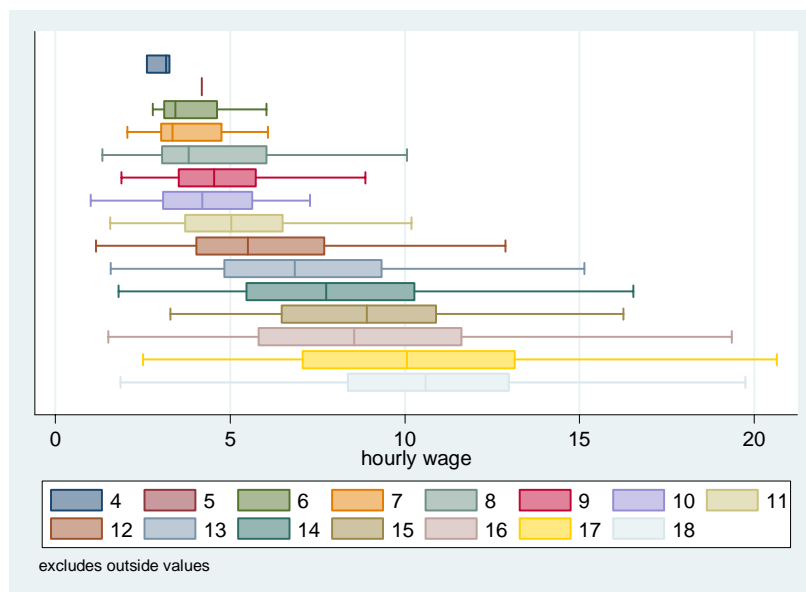
任务 9.10 绘制上学年级与工资收入的箱图，箱图将最低、最高、均值以及 95%分位数在同一个图中表现出来。

***=====begin=====**

sysuse nlsw, clear

```
graph hbox wage, over(grade) asyvar nooutsides legend(rows(2))
```

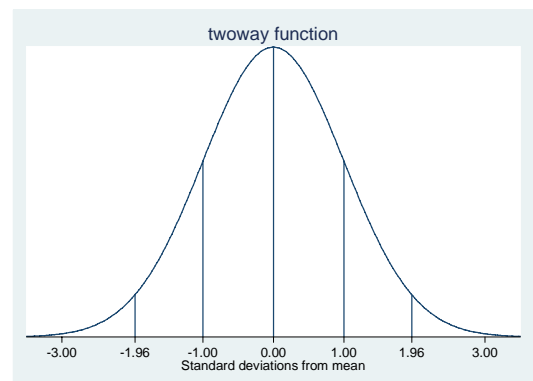
```
*=====end=====
```



9.2.5 函数图

任务 9.11: 绘制如右图的正态分布密度函数图。

```
twoway ///
(function y=normden(x) , range(-3.5 3.5) ///
droplines(-1.96 -1 0 1 1.96)) ///
, ///
title(twoway function) ///
plotregion(margin(zero)) ///
yscale(off) ylabel(, nogrid) ///
xlabel(-3 -1.96 -1 0 1 1.96 3 , format(%4.2f)) ///
xtitle("Standard deviations from mean")
```



9.3 同时做多个图 by(varname)

最常用于描述两个变量之间关系的图形为散点图与拟合图。

```
vguse allstates ,clear
```

```
twoway (scatter propval100 popden) (lfit propval100 popden)(qfit propval100 popden)
```

```
twoway (scatter propval100 popden) (mspline propval100 popden) ///
```

```
(fpfit propval100 popden) (mband propval100 popden)(lowess propval100 popden)
```

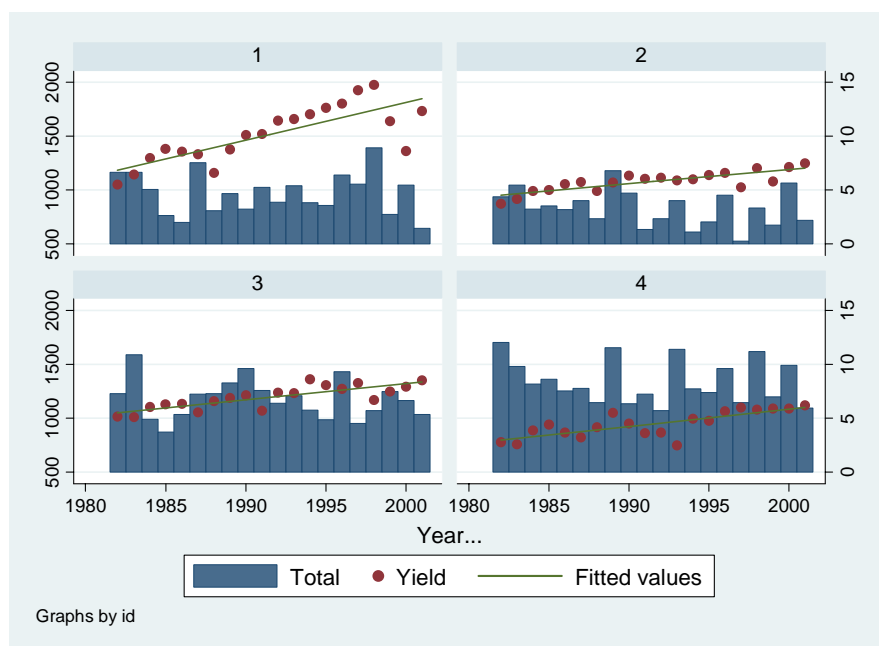
```
twoway (lfitci propval100 popden) (scatter propval100 popden)
```

by(varname)功能适用于 matrix 和 star 以外所有图形,该选项使 graph 按照指定的分组变量分别绘制图形。一般情况下,对 bar 和 box,按指定变量分组的多个图共用一套坐标轴,

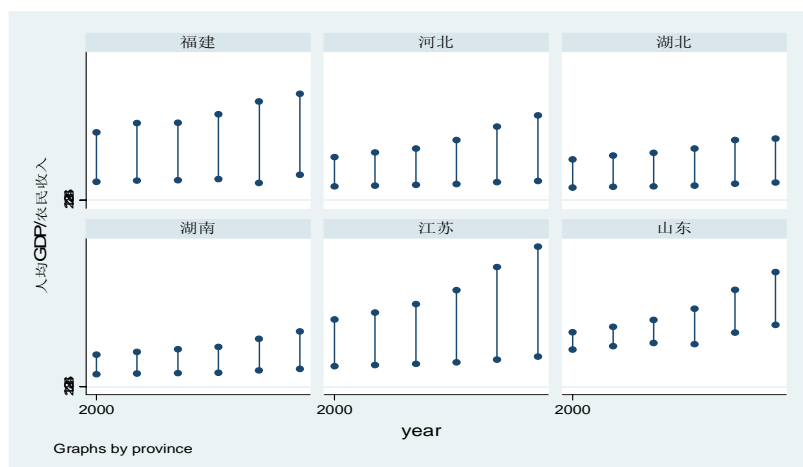
对其它图形，将分别绘制独立的图形并列陈列。**Rescale** 与 **by()**合用，要求图形使用不同的刻度。默认为所有图形使用相同的刻度。

任务 9.12: 利用 **rf.dta** 中的水稻单产 (**yield**) 和总降雨量 (**rainfall**) 以及年份，绘制出单产的散点图和拟合图，同时将总降雨量作为条形图列在另一个坐标轴上，最后，要求同时给出四个县的相应图形。

```
use rf, clear
twoway (bar rainfall year, yaxis(2)) ///
      (scatter yield year, yaxis(1)) ///
      (lfit yield year, yaxis(1)) , ///
      by(id) legend(row(1))
```



任务 7.2: 利用第二次上机收集的 10 个省的 GDP, 人口及农民人均纯收入数据绘制出人均国民收入与农民人均收入的差异图。如下



```
use ourhm, clear
twoway rcapsym ni income year, xlabel(2000(2000)2004) ylabel(0.2(0.6)2.4) by(prov)
```

9.4 模板及图文件处理

(1) 图模板

图的各种设定可以组合成模板，当选定模板后，线性、颜色等各种参数设定相应固定，见下述示例。

sysuse auto, clear

set scheme s2color //see help scheme_s2color

sc mpg weight

set scheme s2mono

sc mpg weight

set scheme s2manual

sc mpg weight

set scheme s1color

sc mpg weight

set scheme s1mono

sc mpg weight

set scheme s1rcolor

sc mpg weight

set scheme s1manual

sc mpg weight

set scheme sj

sc mpg weight

set scheme economist

sc mpg weight

(2) 保存图

graph save "c:\ex7\rf.gph"

graph save "c:\ex7\rf.gph", asis replace

asis将图存为不可再行改动的图，如果不用该选项，则图被存为可修改的图，可以改变其风格和图的大小，在同一文件目录下如果有同名文件，replace选项将其覆盖。

(3) 打开已保存的图

graph use "c:\ex7\rf.gph" [, scheme(s1)]

(4) 显示图

graph display [, scale(1.2) ysize(3) xsize(5) scheme(s1)]

9.5 附录

STATA提供了若干图形示例及代码，这些图例见网址

<http://www.ats.ucla.edu/stat/stata/Library/GraphExamples/default.htm>

10 随机模拟

只要你自己试试模拟随机现象几次，就会加强对概率的了解，比读很多页的数理统计和概率论的文章还有用。学习模拟，不仅是为了解模拟本身，也是为更了解概率而了解模拟。

2000 个人，他们做 1000 次摸球，开始红球白球数目各半，但是如果开始摸对的话，则后面他将要摸到的球正确的概率上升，否则下降。最后结果会如何？

```
clear
mata
R=uniform(1000,2000)
R[1,.]=R[1,.]:>uniform(1,2000)
for (i=2;i<=2000;i++) {
R[i,.]=(uniform(1,2000):>mean(R))
}

(mean(R))'
```

10.1 伪随机数

生成 (0, 1) 之间均匀分布的伪随机数的函数为 `uniform()`

```
di uniform()
```

```
di uniform()
```

```
di uniform()
```

每次都得到一个大于零小于 1 的随机数。

如果要生成一位数的随机数（即 0, 1, 2, 3, 4, 5, 6, 7, 8, 9），可以取小数点后第一位数，通常用下面的命令

```
di int(10*uniform())
```

两位随机数(0-99)则取小数点后两位小数，即

```
di int(100*uniform())
```

任意均匀分布随机数 (a,b) 由下述函数得到

```
a+(b-a)*uniform()
```

任意均匀分布整数随机数 (a,b) 由下述函数得到

```
a+int((b-a)*uniform())
```

也可以同时生成多个随机数，然后将该随机数赋给某个变量。要注意的是，电脑中给出的随机数不是真正的随机数，而是伪随机数，因为它是按照一定的规律生成的。如果给定基于生成伪随机数的初始数值（即 `set seed #`），则对相同的初始数值，生成的伪随机数序列完全一样。

```
*=====begin=====
```

```
clear
```

```
set obs 10
```

```
gen x1=uniform()
```

```
gen x2=uniform() //注意到 x1 与 x2 不一样
```

```
set seed 1234
```

```
gen y1=uniform()
```

```
set seed 1234
```

```

gen y2=uniform()
gen y3=uniform()    //注意到 y1 与 y2 一样，但均与 y3 不同
set seed 5634
gen z1=uniform()
set seed 1234
gen z2=uniform()    //注意到 z2 与 y1, y2 一样，但 z1 与 z2 不同
list
=====end=====

```

可以从 STATA 上下载专门用于生成服从若干主要分布的随机数生成命令。

```

search rnd, net
rndbin 1000 0.5 1    //生成 1000 个 0-1 分布，取 1 的概率为 0.5 (抛均匀的硬币)

```

10.2 简单模拟

利用随机数字表或者电脑软件中的随机数字，来模仿机遇现象，叫模拟（simulation）、

一旦有了可靠的概率模型，模拟是找出复杂事件发生概率的有效工具。一个事件在重复结果中发生的比例，迟早会接近它的概率，所以模拟可以对概率做适当的估计。

例 1：如何执行模拟

掷一枚硬币 10 次，结果中会出现至少 3 个连续正面或者至少 3 个连续反面的概率是多少？

思考：

- (1) 猜想这个概率大约是多少？
- (2) 如何从理论上计算出这个概率？
- (3) 如何模拟计算这个概率？

第一步：提出概率模型。

- 每一次掷，正面和反面的概率各为 0.5
- 投掷之间，彼此是独立的。也就是说，知道某一次掷出的结果，不会改变任何其他次所掷结果的概率。

第二步：分配随机数字以代表不同的结果。

- 随机数字表中的 0-9 每个数字出现的概率都是 0.1
- 每个数字模拟掷一次硬币的结果。
- 奇数代表正面，偶数代表反面。

第三步：模拟多次重复。

- 生成 10 个随机数字
- 记录开心的事件（至少连续三个正面或反面）是否发生，如果发生，记为 1，否则为 0
- 重复 10 次（或者 100，1000，1000000 次），计算概率=开心事件发生/总重复次数。

真正的概率是 0.826。

大部分的人认为连续正面或反面不太容易发生。但模拟结果足以修正我们直觉错误。

```

=====begin=====
cap prog drop seq3
prog seq3, rclass    //rclass 选项表示计算结果将由 return 返回到 r()
version 9
drop _all            //清空所有数据，不能用 clear
set obs 10            //将生成 10 个观察值

```

```

tempvar x y z           //设定 x, y, z 为临时变量
gen `x'=int(10*uniform()) //产生 10 个随机变量, 可能为 0, 1, ..., 9
gen `y'=(mod(`x',2)==0)  //如果生成的随机变量为奇数, 则 y=0; 为偶数, y=1
gen `z'=0                //生成 Z=0
forvalues i=3/10 {
    replace `z'=1 if `y'==`y'[_n-1] & `y'==`y'[_n-2] in `i' //连续三个变量相等时 z=1
}
sum `z'
return scalar max=r(max) //z 取 1 表示高兴的事发生 (三连续), 否则失败
end
simulate max=r(max),reps(10000) nodots:seq3 //重复 1 万次, 取平均结果
sum
*=====end=====

```

另一种更简单, 但不尽规范的程序是:

```

clear
set obs 10           //将生成 10 个观察值
gen y=0
gen z=0
capt prog drop seq3
prog seq3
replace y=uniform()>0.5 //如果生成的随机变量为奇数, 则 y=0; 为偶数, y=1
replace z=0
forvalues i=3/10 {
    replace z=1 if y==y[_n-1] & y==y[_n-2] in `i'
}
sum z
end
simulate r(max),reps(10000) nodots:seq3 //重复 1 万次, 取平均结果
sum

```

由于上述命令要不停生成变量, 进行变量代换, 所以计算速度较慢, 如果使用矩阵, 则计算速度会大大加快, 命令也更简捷。

```

capt prog drop seq3
program seq3
drop _all           //清空所有数据, 不能用 clear
rndbin 10 0.5 1
gen z=0             //生成 Z=0
forvalues i=3/10 {
    replace z=1 if xb==xb[_n-1] & xb==xb[_n-2] in `i' //连续三个变量相等时 z=1
}
sum z
end
simulate max=r(max),reps(10000) nodots:seq3 //重复 1 万次, 取平均结果

```

```
sum
```

```
*MATA
```

```

=====begin=====
mata
A=uniform(10000,10):>0.5 //每行为一次试验，每列为某次试验中硬币的正反面结果
B=J(10000,1,0) //记录每次试验的结果，先记为 0。若高兴结果出现再改为 1
for (j=1;j<=rows(A);j++) { //第 j 次试验
    for (i=3;i<=cols(A);i++) { //第 j 次试验中第 i 次硬币出现结果
        if ( A[j,(i-2,i-1,i)]==(0,0,0) | A[j,(i-2,i-1,i)]==(1,1,1) ) {
            B[j,1]=1 //若连续为正面或者连续为反面，则记为 1
            break
        }
    }
}
mean(B) //求开心事件的次数
end
=====end=====

```

10.3 复杂模拟

例 2：我们要女儿

任务：一对夫妇计划生孩子生到有女儿才停，或者生了三个就停，他们拥有女儿的概率是多大？

思考：理论上，概率是多少？

第一步：概率模型

每一个孩子是女孩的概率是 0.49，且各个孩子的性别是互相独立的。

第二步：分配数字

00, 01, 02, . . . , 49=女孩

49, 50, 51, . . . , 99=男孩

第三步：模拟

从随机数表中生成一对一对的数字，直到有了女儿，或已有 3 个孩子。

重复 100 次。

6905 16 48 17 8717 648987

男女 女 女 女 男女 男男男

用数学可以计算出来，有女孩的真正概率是 0.867

```

=====begin=====
capt program drop girl
program girl, rclass
drop _all
set obs 3
gen x=int(100*uniform())
gen y=(x<49) //生女孩 y=1, 生男孩, y=0

```

```

sum y
return scalar max=r(max)    //若至少有一个女孩，则 max 取 1，若三个全为男孩，取 0
end
simulate max=r(max),reps(10000) nodots:girl
sum
*=====end=====

```

另一种办法

```

clear
set obs 3
gen y=0
capt program drop girl
program girl
replace y=(int(100*uniform())<49)    //生女孩 y=1, 生男孩, y=0
sort y
scalar girl=y[_N]    //若至少有一个女孩，则 girl 取 1，若三个全为男孩，取 0
end
simulate girl, reps(10000) nodots:girl
sum

```

更快的模拟方式。

```

*=====begin=====
capt program drop girl
program girl
mat A=matuniform(1,3)
scalar girl=1
    if A[1,1]>0.49 & A[1,2]>0.49 & A[1,3]>0.49 {
        scalar girl=0    //若三个全为男孩，生女孩数为 0
    }
end
simulate girl, reps(10000) nodots:girl
tab _sim
*=====end=====

*=====begin=====
mata
A=uniform(10000,3):<0.49
B=J(10000,1,1)
for (i=1;i<=rows(A);i++) {
    if (A[i,]==(0,0,0)) {
        B[i,1]=0    //若三个全为男孩，生女孩数为 0
    }
}

```



```

    }
  }
  mean(B)
end
*=====end=====

```

10.4 多阶段模拟

例 3：肾脏移植

张三肾脏不行了，正在等待换肾，他的医师提供了和他情况类似的病人资料。“换肾后的五年存活率：撑过手术的有0.9，术后存活的人中有0.6移植成功，0.4还得回去洗肾；五年后，有新肾的人70%仍然活着，而洗肾的只有一半仍活着。”张三希望知道，他能活过五年的概率，然后再决定是否换肾。

思考：计算理论概率

第一步：采用概率树将信息组织起来。

第二步：分配数字

阶段1：

0=死亡

1-9=存活

阶段2：

0-5=移植成功

6-9=仍需洗肾

阶段3，成功

0-6=存活五年

7-9=死亡

阶段3：洗肾

0-4=存活五年

5-9=死亡

第三步：模拟

数学计算结果为0.558。

```

*=====begin=====

/*换肾后的五年存活率：撑过手术牛0.9，术后存活的人中有0.6移植成功，0.4还得回去
洗肾；五年后，有新肾的人70%仍然活着，而洗肾的只有一半仍活着。计算换肾的人活过五
年的概率。*/

```

```

capt program drop surv
program surv,rclass
drop _all
set obs 1
gen z=1
gen x1=int(10*uniform())
if x1==0 {
    replace z=0
}

```

```

else {
    gen x2=int(10*uniform())
    if x2<6 {
        gen x3=int(10*uniform())
        if x3>6 {
            replace z=0
        }
    }
    else {
        gen x4=int(10*uniform())
        if x4>4 {
            replace z=0
        }
    }
}

sum z
return scalar max=r(max)
end

simulate max=r(max),reps(10000) nodots: surv
sum

```

```

*=====end=====

```

更简捷的方式

```

*=====begin=====

capt program drop surv
program surv
    scalar z=1
    if uniform()<0.1 {
        scalar z=0
    }
    else {
        if uniform()<0.6 {
            if uniform()>=0.7 {
                scalar z=0
            }
        }
        else {
            if uniform()>=0.5 {
                scalar z=0
            }
        }
    }
end

simulate z,reps(10000) nodots: surv

```

sum

*=====end=====

10.5 商店案例

任务：设某种商品每周的需求量 X 是 $[10, 30]$ 上均匀分布的随机变量，而某店进货数量为 $[10, 30]$ 中的某一整数。商店每销售一单位商品可获利 500 元；若供大于求则削价处理，每处理一单位商店亏损 100 元；若供不应求则可从外部调剂供应，此时每一单位商店仅获利 300 元。为使商店所获利润期望值不少于 9280 元，试确定最少进货量。

解：由题设，随机变量 X 的概率密度为

$$f(x) = \begin{cases} \frac{1}{20} & 10 \leq x \leq 30 \\ 0 & \text{其它} \end{cases}$$

设商店进货量为 a 则商店利润 Z 为

$$Z = \begin{cases} 500X - 100(a - X) \\ 500a + 300(X - a) \end{cases}$$

$$= \begin{cases} 600X - 100a & 10 \leq X \leq a \\ 300X + 200a & a \leq X \leq 30 \end{cases}$$

$$\therefore E(Z) = \int_{-\infty}^{\infty} Z(x)f(x)dx$$

$$= \int_{-\infty}^{10} 0dx + \int_{10}^{30} \frac{1}{20} Z(x)dx + \int_{30}^{\infty} 0dx$$

$$= \int_{10}^{30} \frac{1}{20} Z(x)dx$$

$$= \frac{1}{20} \int_{10}^a (600x - 100a)dx + \\ + \frac{1}{20} \int_a^{30} (300a + 200x)dx$$

$$= \frac{1}{20} [300x^2 - 100ax]_{10}^a +$$

$$+ \frac{1}{20} [150x^2 + 200ax]$$

$$= -7.5a^2 + 350a + 5250 \geq 9280$$

$$\therefore 7.5a^2 - 350a + 4030 \leq 0$$

$$(3a - 62)(2.5a - 65) \leq 0$$

$$\therefore 20\frac{2}{3} = \frac{62}{3} \leq a \leq \frac{65}{2.5} = 26$$

故知最小进货量为 21 个单位可使期望利润大于 9280 元。

*=====end=====

```

captu program drop goods
program goods
scalar x=10+int(20*uniform())
if `1'>=x {
    scalar z=500*x-100*(`1'-x)
}
else {
    scalar z=500*`1'+300*(x-`1')
}
end

set more off
quietly forvalues i=10/30 {
simulate z,rep(1000) nodots: goods `i'
quietly sum
scalar z`i'=r(mean)
}
scalar list

*=====end=====

```

10.6 练习

亚洲随机甲虫的命运 (Asian stochastic beetle)，这种昆虫的繁殖模式是：(1) 20%的虫还没有生雌幼虫之前就死掉了，30%生 1 只雌虫，50%生 2 只雌虫。(2) 个别雌虫的繁殖情况互相独立。问：亚洲随机甲虫的前途会：繁殖很快？勉强保持数目？逐渐灭绝？

生日问题：只要一间屋里有 23 个人，则至少有两人同一天生日的概率会超过 1/2。试模拟两个班 60 名学生同一天过生日的概率，并用你们班，或者前几届后几届班组验证之。

古罗马时代的赌博：掷 4 块绵羊距骨是古罗马最受欢迎的赌博，掷多次后骨头的四个面挖概率如下：

窄而平的一面	0.1	宽而凹的一面	0.4
宽而凸的一面	0.4	窄而凹的一面	0.1

掷 4 块距骨最好的结果叫“维纳斯”，这是朝上的四个面都不一样的情况，估计掷出“维纳斯”的概率。

中国会有多少人成为可怜的单身汉？ 不少中国人（尤其是农村）有重男轻女思想，他们总是想尽办法生男孩，性别失调可能会导致越来越多的可怜的男性单身汉。假设所有夫妇都直到生出男孩，或者生够 3 个孩子才停止生育，则一个家庭平均会有几个孩子？多少个男孩？多少个女孩？

商店的平均获利：一商店经销某种商品每周进货的数量 X 与顾客对该种商品的需求量 Y 是相互独立的随机变量，且都服从区间[10,20]上的均匀分布。商店每售出一单位商品可获得利润 1000 元，若需求量超过了进货量，商店可从其它商店调剂供应，这时每单位商品可获利润 500 元，试计算商店经销该种商品所获利润的期望值。

$$\begin{aligned}
 f(x, y) &= f_X(x) f_Y(y) \\
 &= \begin{cases} \frac{1}{100} & 10 \leq x \leq 20 \\ & 10 \leq y \leq 20 \\ 0 & \text{其它} \end{cases}
 \end{aligned}$$

中国人民大学 陈传波
chriscb@126.com

设随机变量 Z 表示商店每周所获利润，则

$$\begin{aligned}
 Z &= \begin{cases} 1000Y \\ 1000X + 500(Y - X) \end{cases} \\
 &= \begin{cases} 1000X & Y \leq X \\ 500(X + Y) & Y > X \end{cases} \\
 E(Z) &= \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} Z(x, y) f(x, y) dx dy \\
 &= \iint_{D_1} 1000y * \frac{1}{100} dx dy + \iint_{D_2} 500(x + y) dx dy \\
 &= 10 \int_{10}^{20} dy \int_y^{20} y dx + \int_{10}^{20} dy \int_{10}^y 5(x + y) dx \\
 &= 10 \int_{10}^{20} y(20 - y) dy + \int_{10}^{20} dy \left(\frac{5}{2} y^2 - 10Y - 50 \right) dy \\
 &= 10 \left[10y^2 - \frac{1}{3} y^3 \right]_{10}^{20} + 5 \left[\frac{1}{2} y^3 - 5y^2 - 50y \right]_{10}^{20} \\
 &= \frac{20000}{3} + 5 * 1500 \\
 &\approx 14167
 \end{aligned}$$

一间宿舍中有四位同学的眼镜都放在书架上，去上课时，每人任取一付眼镜，求每个人都没有拿到自己眼镜的概率。

10.7 附录

亚洲随机甲虫的命运 (Asian stochastic beetle)，这种昆虫的繁殖模式是：(1) 20%的虫还没有生雌幼虫之前就死掉了，30%生 1 只雌虫，50%生 2 只雌虫。(2) 个别雌虫的繁殖情况互相独立。问：亚洲随机甲虫的前途会：繁殖很快？勉强保持数目？逐渐灭绝？

```

=====begin=====
capture prog drop bb
prog bb, rclass
local beetle = 1
while `beetle' < 500 & `beetle' > 0 {
drop _all
set obs `beetle'
tempvar x y
gen `x' = uniform()
gen `y' = 1
replace `y' = 2 if `x' < 0.5
replace `y' = 0 if `x' > 0.8
}

```

```

quietly sum `y'
local beetle=r(sum)
}
noisily display as error `beetle'
end

```

```

set more off
quietly bb 10

```

```

*=====end=====

```

生日问题：只要一间屋里有 23 个人，则至少有两人同一天生日的概率会超过 1/2。试模拟两个班 60 名学生同一天过生日的概率，并用你们班，或者前几届后几届班组验证之。

```

*=====begin=====

```

```

captu prog drop birth
prog birth
drop _all
set obs 60
tempvar y
gen `y'=int(365* uniform())
sort `y'
scalar z=0
    forvalues i=1/59 {
        if `y'[`i']==`y'[`i'+1] {
            scalar z=1
            continue, break
        }
    }
end
simulate "birth" z, reps(100)
sum

```

```

*=====end=====

```

古罗马时代的赌博：掷 4 块绵羊距骨是古罗马最受欢迎的赌博，掷多次后骨头的四个面挖概率如下：

窄而平的一面	0.1	宽而凹的一面	0.4
宽而凸的一面	0.4	窄而凹的一面	0.1

掷 4 块距骨最好的结果叫“维纳斯”，这是朝上的四个面都不一样的情况，估计掷出“维纳斯”的概率。

```

*=====begin=====

```

```

captu prog drop wns
prog wns
drop _all
set obs 4
tempvar x y

```

```

gen `y'= uniform()
recode `y' (0/0.1=1) (0.1/0.2=2) (0.2/0.6=3) (0.6/1=4), gen(`x')
sort `x'
scalar z=1
    forvalues i=1/4 {
        if `x'[`i']==`x'[`i'+1] {
            scalar z=0
            continue, break
        }
    }
end
simulate "wns" z, reps(1000)
sum
*=====end=====

```

中国会有多少人成为可怜的单身汉？不少中国人（尤其是农村）有重男轻女思想，他们总是想尽办法生男孩，性别失调可能会导致越来越多的可怜的男性单身汉。假设所有夫妇都直到生出男孩，或者生够3个孩子才停止生育，则一个家庭平均会有几个孩子？多少个男孩？多少个女孩？

```

*=====begin=====
captu prog drop child
prog child
drop _all
set obs 3
tempvar y
gen `y'=int(100*uniform())
scalar boy=0
scalar girl=0
if `y'[1]<49 {
    scalar girl=1
    if `y'[2]<49 {
        scalar girl=2 //第二个为女孩
        if `y'[3]<49 {
            scalar girl=3 //三女
        }
    }
    else {
        scalar boy=1 //2女1男
    }
}
else {
    scalar boy=1 //1女1男
}
}
else {

```

```

                scalar boy=1 //0女1男
            }

end
simulate "child"  boy=boy girl=girl, reps(1000)
sum
=====end=====
如果不是重男轻女，而是生到三个为止，不论是男是女，则
=====begin=====

    captu prog drop child
    prog child, rclass
    drop _all
    set obs 3
    tempvar y b g
    gen `y'=int(100*uniform())
    gen `b'=0
    gen `g'=0
    replace `b'=1 if `y'>=49
    sum `b'
    scalar boy=r(sum)
    replace `g'=1 if `y'<49
    sum `g'
    scalar girl=r(sum)
end
simulate "child"  boy girl, reps(1000)
sum
=====end=====

```

商店的平均获利：一商店经销某种商品每周进货的数量 X 与顾客对该种商品的需求量 Y 是相互独立的随机变量，且都服从区间 $[10, 20]$ 上的均匀分布。商店每售出一单位商品可获得利润 1000 元，若需求量超过了进货量，商店可从其它商店调剂供应，这时每单位商品可获利润 500 元，试计算商店经销该种商品所获利润的期望值。

当 X 和 Y 取连续变量时， $X=[10, 20]$ ， $Y=[10, 20]$

```

=====begin=====

    capture program drop pf
    prog pf
    scalar x=10+10*uniform()
    scalar y=10+10*uniform()
    if x>=y {
        scalar z=1000*y
    }
    else {
        scalar z=500*(x+y)
    }
end
simulate " pf" z, reps(10000)

```



```
sum
```

```
*=====end=====
```

```
结果约为: 14153.12
```

当进货量和销售量均取整数时，即 $x=10, 11, \dots, 20$ ；而 $y=10, 11, \dots, 20$

```
*=====begin=====
```

```
capture program drop pf
prog pf
  scalar x=10+int(11*uniform())
  scalar y=10+int(11*uniform())
  if x>=y {
    scalar z=1000*y
  }
  else {
    scalar z=500*(x+y)
  }
end
simulate " pf" z, reps(10000)
sum
```

```
*=====end=====
```

```
结果约为: 14102.65
```

```
clear
mat A=J(11,11,.)
forvalue i=1/11 {
  forvalue j=1/11 {
    if `i'>=`j' {
      mat A[`i',`j']=1000*(9+`i')
    }
    else if `i'<`j' {
      mat A[`i',`j']=500*(18+`i'+`j')
    }
  }
}
mat list A
mat a=J(1,11,1)
mat C=a*A*a'/121
mat list C
* 15909.091
```

*当进货量和销货量取 1 位小数时

```
clear
mat A=J(101,101,.)
```

```

forvalue i=1/101 {
    forvalue j=1/101 {
        if `i'>=`j'{
            mat A[`i',`j']=1000*(10+(`i'-1)/10)
        }
        else if `i'<`j' {
            mat A[`i',`j']=500*(20+(`i'-1)/10+(`j'-1)/10)
        }
    }
}
mat a=J(1,101,1)
mat C=a*A*a'/10201
mat list C
15841.584
=====end=====

```

请思考，上述程序有什么问题？为什么答案有如此大的差距？

II 分布函数

分布函数、密度函数、分位数及其反函数一览表

名称	分布函数（求概率）	密度函数	反函数
二项	$\text{Binomial}(n,k,p)$	$C_n^k p^k (1-p)^{n-k}$	
正态	$\text{normal}(z)$	$\text{normalden}(z)$ $\text{normalden}(z,s) = \text{normalden}(z)/s$ $\text{normalden}(x,m,s) = \text{normalden}((x-m)/s)/s$	$\text{invnormal}(p)$
对数正态	$\text{lnnormal}(z)$	$\text{lnnormalden}(z)$ $\text{lnnormalden}(z,s) = \text{lnnormalden}(z) - \ln(s)$ $\text{lnnormalden}(x,m,s) = \text{lnnormalden}((x-m)/s)$	
卡方	$\text{chi2}(n,x)$		$\text{invchi2}(n,p)$
卡方分位数	$\text{chi2tail}(n,x) = 1 - \text{chi2}(n,x)$		$\text{invchi2tail}(n,p)$
t 分位数	$\text{ttail}(n,t)$	$\text{tden}(n,t)$	$\text{invttail}(n,p)$
F	$F(n1,n2,f)$	$F\text{den}(n1,n2,f)$	$\text{invF}(n1,n2,p)$
F 分位数	$F\text{tail}(n1,n2,f) = 1 - F(n1,n2,f)$		$\text{invFtail}(n1,n2,p)$

11.1 二项分布

$\text{Binomial}(n,k,p)$ 计算成功概率为 p 的随机事件，在 n 次独立重复试验中，成功次数大于等于 k 次的概率。如出现概率为 0.6 的某随机现象，在六次独立重复试验中，出现 4 次及以上的概率为 .54432。

di Binomial(6,4,0.6)

.54432

如果要计算出，恰好成功 4 次的概率，则需要按如下方式求得

di Binomial(6,4,0.6) - Binomial(6,5,0.6)

.27648

掷一枚硬币二次，出现一次正面的概率为

di Binomial(2,1,0.5) - Binomial(2,2,0.5)

0.5

掷一枚硬币二次，至少出现一次正面的概率为

di Binomial(2,1,0.5)

.75

掷一枚硬币二次，出现两次正面的概率为

di Binomial(2,2,0.5)

.25

掷一枚硬币二次，不出现或者出现正面的概率为

di Binomial(2,0,0.5)

1

11.2 标准正态分布函数

在标准正态分布中，出现小于 -1.96 的随机数的概率是 0.025

```
di normal(-1.96)
```

```
.0249979
```

而出现小于 1.96 的随机数的概率为 0.975

```
di normal(-1.96)
```

```
.9750021
```

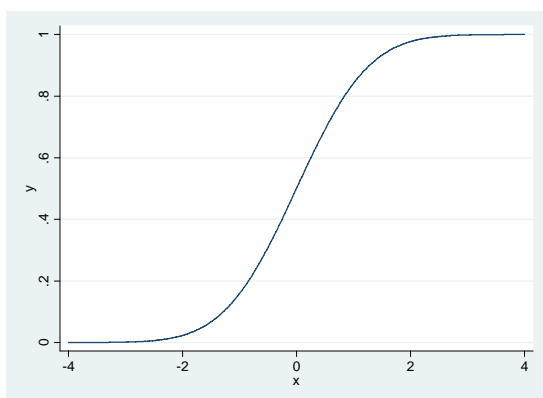
大于 1.96 的随机数出现的概率则为

```
di 1-normal(1.96)
```

```
.0249979
```

标准正态分布函数的图示

```
twoway function y=normal(x), rang(-4 4)
```



任务 11.1：利用计算机得到标准正态分布概率表（p751-753）。

```

=====begin=====
mat z=J(61,11,)
forvalues i=1/61{
    mat z[`i',1]=(`i'-31)/10
    forvalues j=2/11{
        mat z[`i',`j']=normal((`i'-31)/10+(`j'-2)/100)
    }
}
matrix colnames z = z 0 1 2 3 4 5 6 7 8 9
mat list z, format(%5.4f)
=====end=====

```

11.3 正态分布函数及其反函数

一般的正态分布函数，可以根据公式 $(x-m)/s=z$ 来变形得到

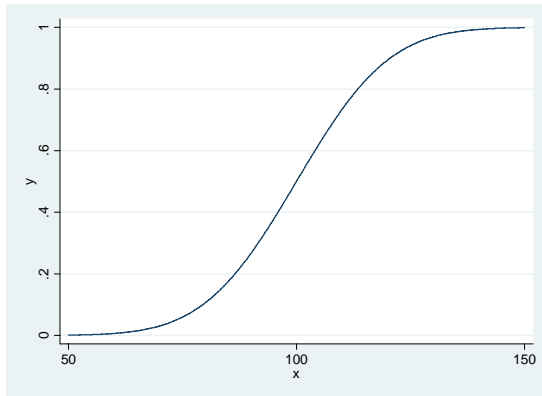
例：人的智商（I.Q.）得分一般服从均值为 100，标准差为 16 的正态分布，随机抽取一人，他的智商在 100-115 之间的概率是多少？（以频率为表述，即智商在 100-115 之间的人占多大比例？）

```
di normal((115-100)/16)- normal((100-100)/16)
```

```
.32574929
```

正态分布函数的图示

```
twoway function y=normal((x-100)/16), rang(50 150)
```



求标准正态分布累积函数值为 0.975 的点对应的随机数

```
di invnormal(.975)
```

```
1.959964
```

结果为 1.96，正好与 `normal(1.96)` 相对应，他们互为反函数。类似地计算

```
. di invnormal(.995)
```

```
2.5758293
```

例：设在注册会计师的会计科目考试中，其通过率只有 10%，从历年的经验来看，分数的均值和标准差分别为 72 和 13。如果分数近似正态分布，为了获得顶部 10% 的分数并通过考试所需要的最小分数是多少？

```
di invnormal(0.9)*13+72
```

```
88.66017
```

11.4 服从正态分布的随机数

定理：设 X 是一个连续型随机变量，其分布函数 $F(x)$ 是严格单调递增的，则 $Y=F(X)$ 服从 $[0, 1]$ 上的均匀分布。(P64)

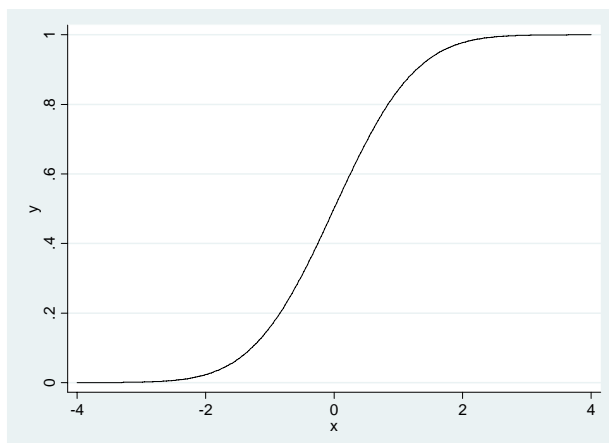
证明：由于 $y=F(x)$ 是严格单调递增函数，因而其反函数存在，记为 $x=invF(y)$ 。

Y 的分布函数为 $P(Y \leq y) = P(F(X) \leq y)$

由于 Y 的值域为 $[0, 1]$ ，故当 $y < 0$ 时， $p=0$ ；当 $y > 1$ 时， $p=1$ ，当 $0 \leq y \leq 1$ 时，

$$P(F(x) \leq y)$$

$$= p\{invF(F(x)) \leq invF(y)\} = p\{x \leq invF(y)\} = F(invF(y)) = y$$



利用上述性质，可以很容易地得到满足各种分布的随机数。如标准正态分布的随机数为

```
di invnorm(uniform())
```

例：

```
*=====begin=====
clear
set obs 10000
gen z=invnormal(uniform()) //得到服从标准正态分布的随机数
hist z,bin(100) norm      //画出直方图并配上标准正态分布曲线
*=====end=====
```

11.5 正态分布密度函数

STATA 提供了三种计算正态密度函数的命令，分别是标准正态密度函数

```
di normalden(1.95)
```

```
.05959471
```

均值为零，标准差为 s 的正态密度函数，经换算，与标准正态密度函数等价
 $\text{normalden}(z,s) = \text{normalden}(z)/s$

```
di normalden(1.95,10)
```

```
.00595947
```

均值为 m ，标准差为 s 的正态密度函数，经换算，与标准正态密度函数等价
 $\text{normalden}(x,m,s) = \text{normalden}((x-m)/s)/s$

```
di normalden(29.5,10,10)
```

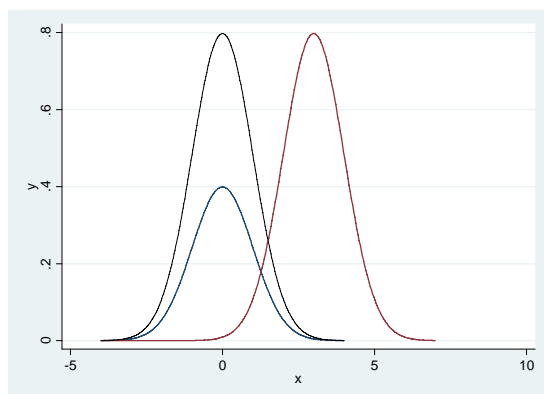
```
.00595947
```

正态分布密度函数的图示

```
*=====begin=====
#delimit ; //表示每个分号为一个命令，不论是否分行
twayay
function y=normden(x), range(-4 4) || /*标准正态密度函数*/
function y=2*normden(x-3), range(-4 7) || /*均值为 3，标准差为 2 的正态
密度函数*/
function y=2*normden(x), range(-4 4) clstyle(foreground) //标准差为 2
legend(off);
*=====end=====
```

#delimit cr

*=====end=====



*=====begin=====

#delimit ;

twoway

function y=normden(x), range(-4 -1.96) bcolor(gs12) recast(area) ||

function y=normden(x), range(1.96 4) bcolor(gs12) recast(area) ||

function y=normden(x), range(-4 4) clstyle(foreground) ||,

plotregion(style(none))

yscale(off) xscale(noline)

legend(off)

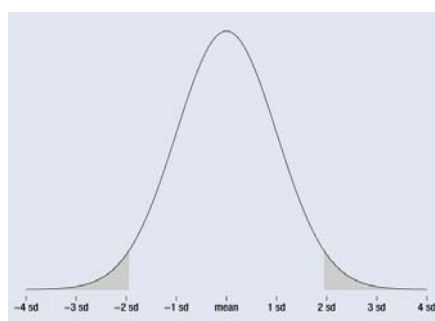
xlabel(-4 "-4 sd" -3 "-3 sd" -2 "-2 sd" -1 "-1 sd" 0 "mean"

1 "1 sd" 2 "2 sd" 3 "3 sd" 4 "4 sd", grid gmin gmax)

xtitle("");

#delimit cr

*=====end=====



11.6 分位数

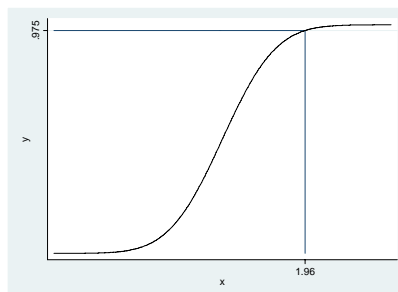
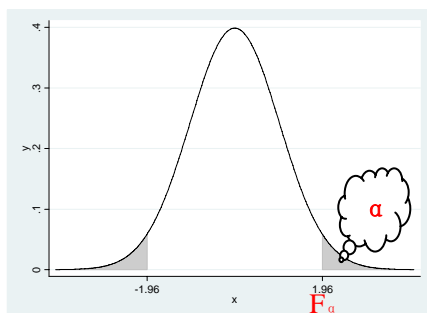
分位数是统计分布的一类数字特征。

定义：设随机变量 X 的分布函数 $F(x)$ ，对给定的实数 α ($0 < \alpha < 1$)，如果实数

F_α 满足：

$P\{X > F_\alpha\} = \alpha$, 即 $1 - F(F_\alpha) = \alpha$, 或者 $F(F_\alpha) = 1 - \alpha$

则称为随机变量 X 的分布的水平 α 的上侧分位数。或分布函数 $F(x)$ 的水平 α 的上侧分位数。 $F_\alpha = \text{invF}(1 - \alpha)$



```

=====begin=====
#delimit ;
twoway
    function y=0.975, rang(-4 1.96) dropline(1.959)||
    function y=normal(x), range(-4 4) clstyle(foreground) ||,
    legend(off)
xlabel(1.96)
ylabel(.975)
;
#delimit cr
=====end=====

```

标准正态分布的水平 $\alpha = 0.05$ 的上侧分位数为

```
di invnormal(0.95)
```

1.6448536

标准正态分布的水平 $\alpha = 0.05$ 的双侧分位数为

```
di invnorm(0.975)
```

1.959964

11.7 卡方分布

卡方分布：设 X_1, X_2, \dots, X_n 是 n 个相互独立的随机变量，且 X_i 均服从标

准正态分布，则 $X = \sum X_i^2$ 服从自由度为 n 的卡方分布。

```

=====begin=====
#delimit;
tw function y=(chi2(2,x)-chi2(2,(x-0.01)))/0.01,rang(0 30) ||
function y=(chi2(4,x)-chi2(4,(x-0.01)))/0.01,rang(0 30) ||
function y=(chi2(8,x)-chi2(8,(x-0.01)))/0.01,rang(0 30),legend(off);
tw function y=chi2(2,x),rang(0 30) ||

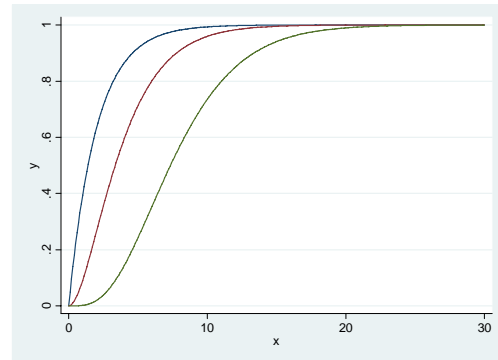
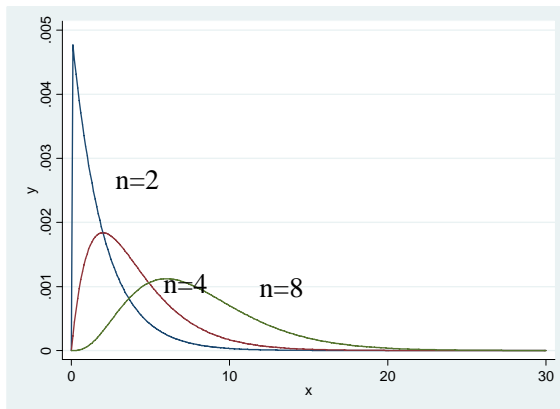
```



```
function y=chi2(4,x),rang(0 30) ||
function y=chi2(8,x),rang(0 30), legend(off);
```

***非中心化卡方分布图

```
tw function y=100*(chi2(2,x)-chi2(2,(x-0.01))),rang(0 30) ||
function y=100*(nchi2(2,4,x)-nchi2(2,4,(x-0.01))),rang(0 30) ||
function y=100*(nchi2(2,8,x)-nchi2(2,8,(x-0.01))),rang(0 30),legend(off);
*=====end=====
```



自由度为10时，累积分布为0.95所对应的随机变量为，即10个独立的标准正态分布随机变量平方和小于18.31的可能性为0.95。

```
di invchi2(10,0.95)
18.307038
di chi2(10,18.31)
.95004583
```

卡方分布的分位数函数与累积分布函数的关系是 $\text{chi2}(n,x)=1-\text{chi2tail}(n,x)$

```
di chi2tail(10,18.31)
.04995417
```

卡方分布的分位数的反函数与其累积分布的反函数的关系是 $\text{invchi2}(n,p)=\text{invchi2tail}(n,1-p)$

```
di invchi2tail(10,0.05)
18.307038
```

任务 11.2：自己做出卡方分布的临界值（P758）

```
*=====begin=====
mat X=J(31,3,.)
forvalues n=1/30{
    mat X[`n',1]=invchi2tail(`n',0.1)
    mat X[`n',2]=invchi2tail(`n',0.05)
    mat X[`n',3]=invchi2tail(`n',0.01)
}
mat list X, format(%5.2f)
*=====end=====
```

11.8 t 分布的分位数

t 分布是标准正态分布与自由度为 n 的卡方分布的函数
自由度为 8 的 t 分布的水平 0.05 的上侧分位数为

```
di invttail(8,0.05)
1.859548
```

即

```
di ttail(8,1.86)
.04996531
```

由于 t 分布为对称分布，因此双侧分位数为

```
di invttail(8,0.025)
2.3060041
di ttail(8,2.306)
.02500016
```

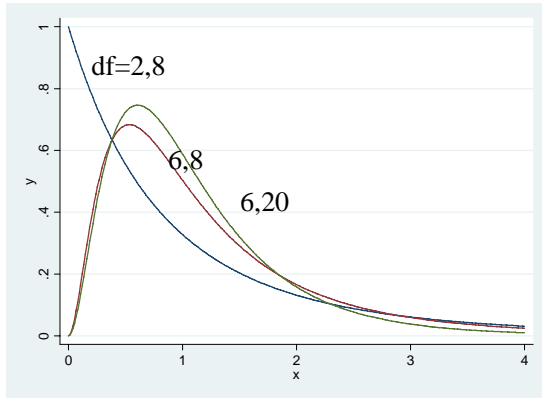
任务 11.3：自己做出 t 分布表 (p754)

```
*=====begin=====
mat t=J(31,5,.)
forvalues n=1/30{
    mat t[`n',1]=invttail(`n',0.1)
    mat t[`n',2]=invttail(`n',0.05)
    mat t[`n',3]=invttail(`n',0.025)
    mat t[`n',4]=invttail(`n',0.01)
    mat t[`n',5]=invttail(`n',0.005)
}
mat list t, format(%5.3f)
*=====begin=====
```

11.9 F 分布

F 分布是两个卡方分布的均商（自由度平均）

```
*=====begin=====
#delimit ;
twoway
function y=Fden(2,8,x), rang(0 4) ||
function y= Fden(6,8,x), rang(0 4) ||
function y= Fden(6,20,x), rang(0 4) legend(off);
*=====begin=====
```



设 x 服从分子自由度为 10，分母自由度为 5 的 F 分布，求 x 小于 4.74 的概率

```
di F(10,5,4.74)
```

```
.95010421
```

```
di invF(10,5,0.95)
```

```
4.7350631
```

```
di Ftail(10,5,4.74)
```

```
.04989579
```

```
di invFtail(10,5,0.05)
```

```
4.7350631
```

设 x 服从分子自由度为 5，分母自由度为 10 的 F 分布，求 x 小于的概率

```
. di F(5,10,3.326)
```

```
.95000672
```

```
. di invF(5,10,0.95)
```

```
3.3258345
```

```
. di Ftail(5,10,3.326)
```

```
.04999328
```

```
. di invFtail(5,10,0.05)
```

```
3.3258345
```

可见， F 分布的累积分布函数与分位数的关系为

$$F(n,m,f)=1-Ftail(n,m,f)$$

反函数之间的关系为

$$invF(n,m,p)=invFtail(n,m,1-p)$$

交换第一自由度与第二自由度，则两个分数之间的关系为

$$Ftail(m,n,a)=1/[Ftail(n,m,1a)]$$

```
di invFtail(5,10,0.05)
```

```
3.3258345
```

```
di 1/(invFtail(10,5,0.95))
```

```
3.3258345
```

任务 11.4：自己做出 F 分布表 (p756)

```
*=====begin=====
```

```
mat F=J(21,11,.)
```

```
forvalues i=1/21{
```

```
mat F[`i',1]=`i'+9
      forvalues j=2/11 {
        mat F[`i',j]=invFtail((`j'-1),(`i'+9),0.05)
      }
}
matrix colnames z = F 1 2 3 4 5 6 7 8 9 10
mat list F, format(%4.2f)
*=====end=====
```

12 抽样分布

本次上机的目的在于通过模拟帮助我们理解总体及其分布；统计理及其抽样分布；由于模拟的样本容量不可能无穷大，因此这里的直方图给出的只能是逼近理论分布的经验分布。

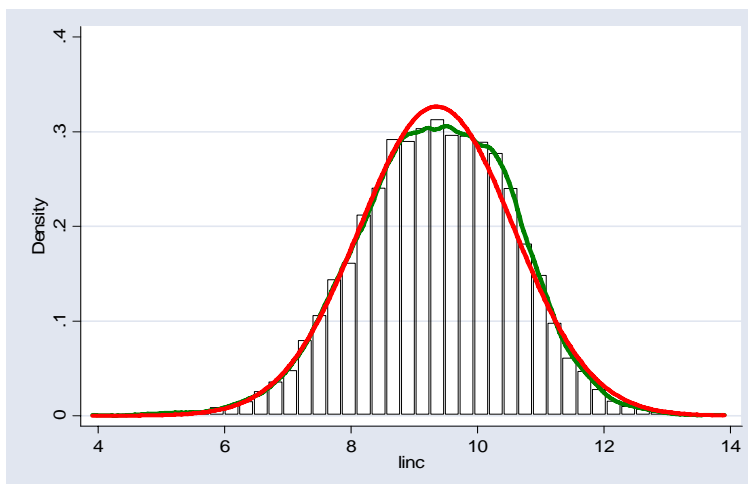
12.1 经验分布

数据 `realinc` 是来自农村固定观察点的农户收入数据（共 2 万多户，仅仅将这些数据列出，并不能得出任何有价值的信息和推断。我们首先绘出该收入的直方图，从图中可以看出，低收入占绝大多数，高收入者少，图形与正态分布有较大的偏差，它有一个长长的右拖尾。接着，我们将该收入取对数，绘出收入对数的直方图，则可以发现，收入对数近似服从正态分布。

```

=====begin=====
use realinc, clear
hist inc, bin(200) norm /*绘出收入直方图，bin(200)表示列出 200 个直方
条。norm 加上标准正态曲线作为参照*/
g linc=ln(inc) //对数变换
hist linc, bin(200) norm //对数变换后的直方图
sum linc, d //求出总体均值和方差，作为参数
g f=exp(-(linc-9.35627)^2/(2*1.221415^2))/(sqrt(2*_pi)*1.221415) //密度函数
*g f=normden(linc, 9.35627, sqrt(1.221415))
line f linc, sort //绘出密度函数曲线
tw hist linc || line f linc, sort //将直方图和理论密度函数绘制在一起
hist linc, bfc color(none) bl color(none) bl width(none) normal
normopts( cl color(red) cl width(thick)) k density k denopts( cl color(green)
cl width(thick) )
=====end=====

```



12.2 均值的抽样分布：正态总体的小样本抽样分布

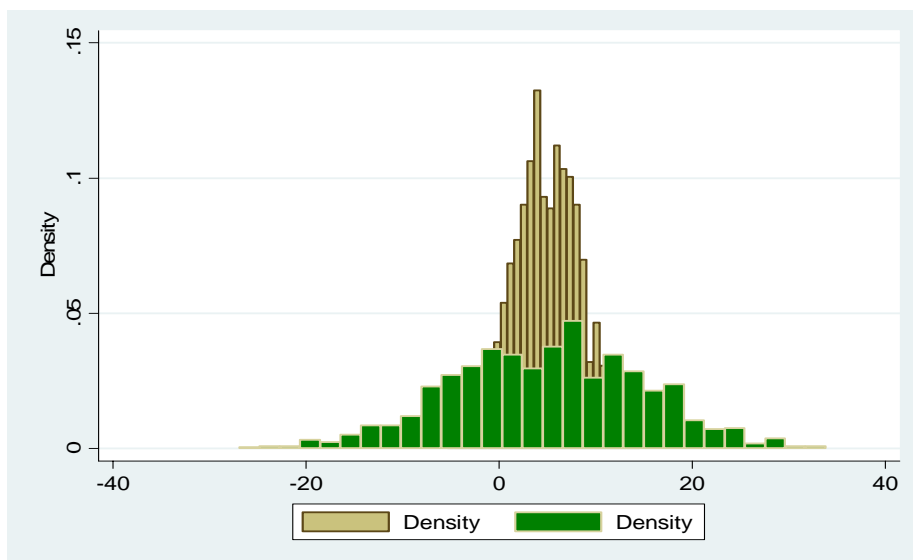
总体指某随机变量（如身高），总体分布一般服从一定的规律（如正态分布）。

抽样分布是统计量所服从的分布，而统计量是样本的函数，如均值为所有样本的算术平均。当多次抽样时，每次抽样所计算出来的平均值也会服从某种分布。如来自正态总体 $N(u, \sigma^2)$ 的均值服从 $N(u, \sigma^2/n)$ （即抽样分布）。下面的例题，首先生成一个均值为 5，标准差为 10 的正态总体，从该正态总体中第一次抽取 8 个样本，计算其均值得 x_1 。然后放回，重新抽取 8 个样本，计算均值得 x_2 ，依次进行 1000 次抽样，得到 1000 个均值，做这些均值的直方图，可以发现，它服从正态分布。

```

=====begin=====
cap prog drop sd
prog sd
drawnorm x,n(8) m(5) sds(10) clear //8 个 u=5, o=10 的正态随机样本
quietly sum x
end
***将上述抽样试验进行 1000 次，得到 1000 个均值和标准差
simulate sd m=r(mean), reps (1000)
drawnorm x, m(5) sds(10)
sum //比较两者的均值和标准差。
tw (hist m, density blcolor(olive) ) (hist x, density bfcolor(green))
=====end=====

```



上图表明，服从正态分布的总体 X （绿色），其抽样分布仍服从正态分布，均值相同（均为 5），但标准差只有原标准差的 $1/\sqrt{8}$ （因为样本为 8）。

12.3 中心极限定理：非正态总体大样本下均值的抽样分布

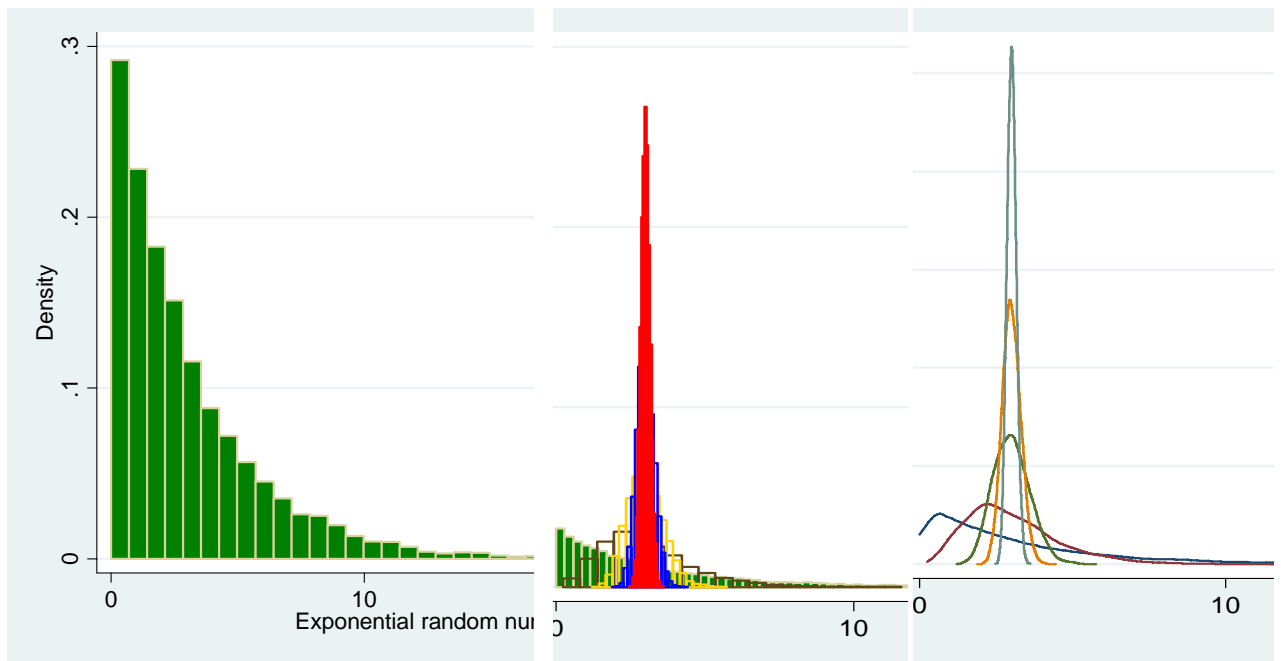
如果原总体不服从正态分布，其均值的抽样分布是否仍为正态分布？

```

=====begin=====
cap prog drop lnsd
prog lnsd    //自建命令 lnsd，从指数分布中抽取指定的样本计算均值
drop _all
rndexp `1' 3    //生成服从指数分布的随机样本，先下载该命令 net get rnd
quietly sum xe
end
*进行四组试验，样本容量分别为 4，25，100，400，每组试验重复 1000 次。
foreach s of numlist 4 25 100 400 {
    simulate m`s'=r(mean), reps (1000) nodots: lnsd `s'
    save temp`s', replace
}
rndexp 1000 3    //生成服从指数分布的随机样本，即原始数据生成机制 DGP
merge using temp4 temp25 temp100 temp400
tw (hist xe, bfccolor(green) bin(100)) ///
    (hist m4, blcolor(olive) bfccolor(none) bin(20)) ///
    (hist m25, blcolor(gold) bfccolor(none) bin(20)) ///
    (hist m100, blcolor(blue) bfccolor(none) bin(20)) ///
    (hist m400, blcolor(red) bfccolor(none) bin(20)) , legend(off)

tw (kdensity xe) (kdensity m4) (kdensity m25) ///
    (kdensity m100) (kdensity m400)
=====end=====

```



最左边的图为指数分布图，中间的图为从左边的指数分布图中依次抽取 4、25、100 和 400 个样本得到的抽样分布直方图，最右边的图则为相应的密度拟合图。从上图可以直观地看出，服从指数分布的总体（绿色），其抽样分布（近似正态）不同于原总体分布（指数分布）。

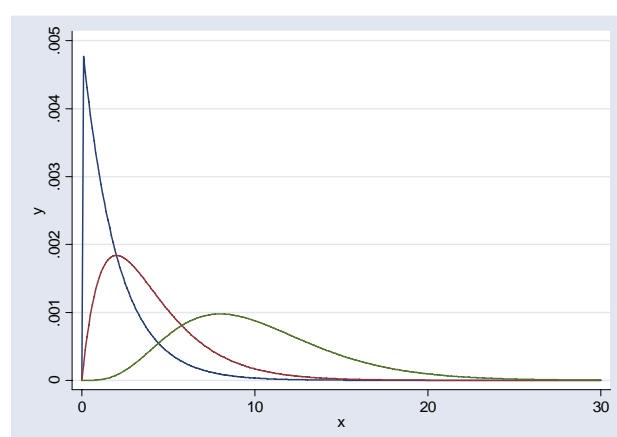
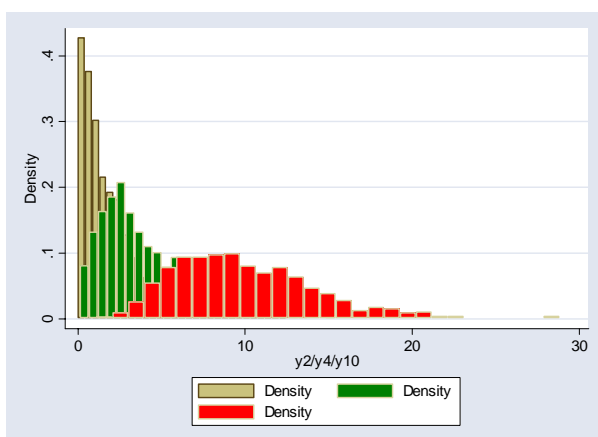
12.4 卡方分布与样本标准差的抽样分布

卡方分布就其定义来看，是标准正态分布的平方和。而抽样分布理论告诉我们，样本方差的一个变形服从卡方分布。下面的程序从标准正态分布出发构建一个卡方分布，然后用卡方分布的密度公式直观地验证卡方分布来源于标准正态分布。

```

=====begin=====
drawnorm x1-x10,n(1000) clear //生成 10 个服从标准正态分布的随机样本
forv i=1/10 {
  replace x`i'=x`i'^2 //将这 10 个样本平方
}
forv i=1/10 {
  egen y`i'=rsum(x1-x`i') //求出所有的平方和，即 y1=x1, y2=x1+x2...
}
tw (hist y2, blcolor(olive) ) (hist y4, bfcolor(green)) (hist y10, bfcolor(red))
tw function y=100*(chi2(2,x)-chi2(2,(x-0.01))),rang(0 30) || ///
function y=100*(chi2(4,x)-chi2(4,(x-0.01))),rang(0 30) || ///
function y=100*(chi2(10,x)-chi2(10,(x-0.01))),rang(0 30) legend(off)
=====end=====

```



我们通过模拟简单随机抽样来理解样本的方差的 $(n-1)$ 倍除以总体方差后服从卡方分布。当 X 服从正态分布时， $(x-u)/\sigma$ 服从标准正态分布，其平方和便服从自由度为 n 的卡方分布，但当用样本均值替代总体均值后，损失掉一个自由度，于是，它服从如下的分布。

$$\frac{n-1}{\sigma^2} S^2 \sim \chi^2(n-1) \quad (2)$$

```

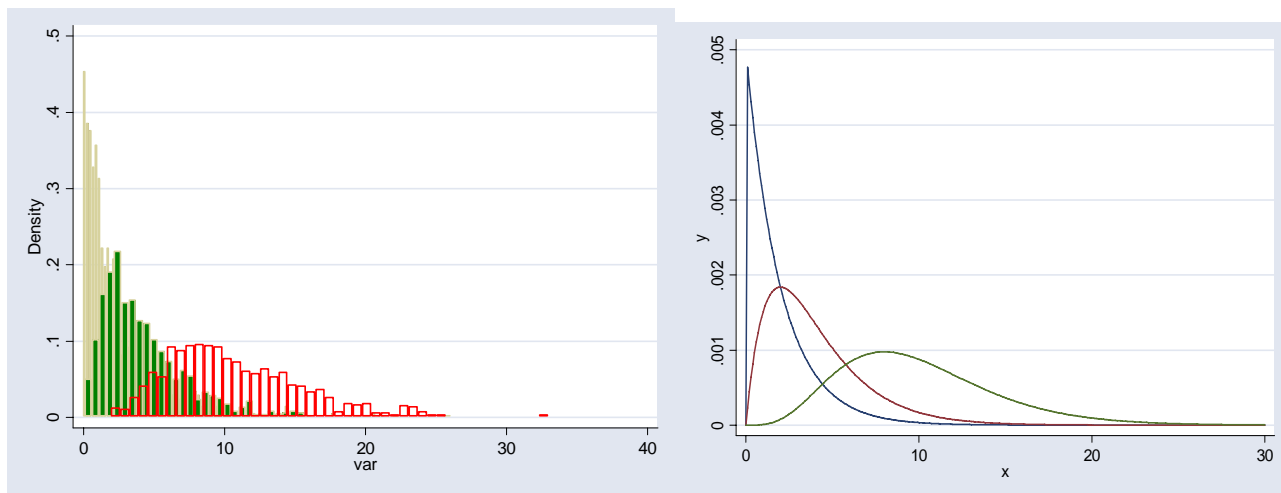
=====begin=====
capt prog drop s
prog s
drawnorm x,n(1') m(5) sds(10) clear
quietly sum x
scalar s=(1'-1)*r(Var)/100 //var是服从卡方分布的随机变量
end

```


*分别进行样本容量为3, 5和11的三组模拟, 每次模拟1000次,

```
foreach i of numlist 3 5 11 {
    simulate var`i'=s, reps (1000) nodots: s `i'
    save temp`i', replace
}
merge using temp3 temp5
tw (hist var3, bfcolor(olive)) (hist var5, bfcolor(green)) ///
(hist var11, blcolor(red) bfcolor(none))
```

*=====begin=====



12.5 构造 F 分布

服从自由度为 n 的卡方分布与服从自由度为 m 的卡方分布, 两者分别除以其自由度后的比例服从 F 分布。下面的程序构造了 F 分布, 并相应地画出了其直方图和函数曲线。

*=====begin=====

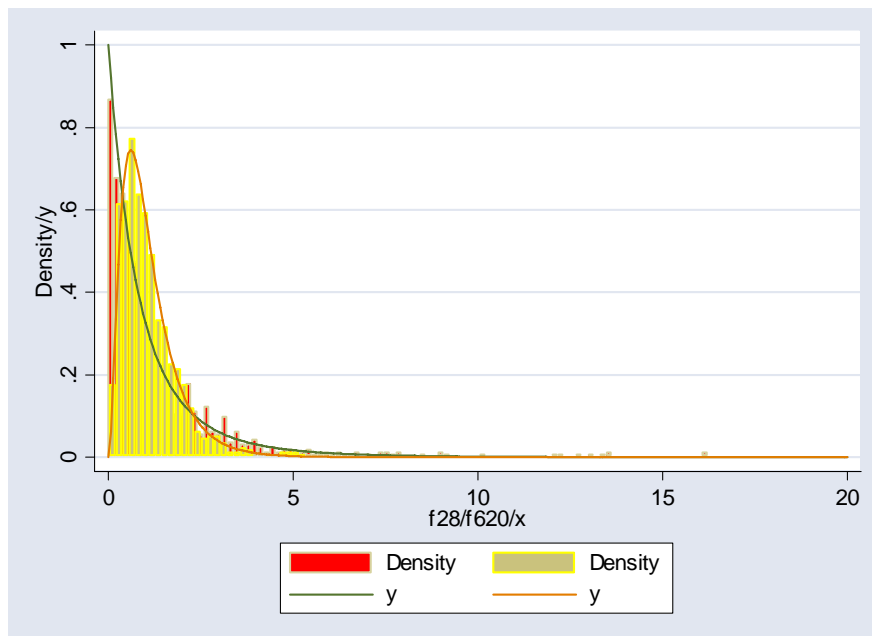
```
drawnorm x1-x20,n(1000) clear
forv i=1/20 {
    replace x`i'=x`i'^2
}
egen y8=rsum(x1-x8)
egen y20=rsum(x1-x20)
keep y8 y20
```

```
drawnorm x1-x10,n(1000)
forv i=1/10 {
    replace x`i'=x`i'^2
}
egen z2=rsum(x1-x2)
egen z6=rsum(x1-x6)
keep y* z2 z6
```

```

g f28=(z2/2)/(y8/8)
g f620=(z6/6)/(y20/20)
tw hist f28,bin(100) bfcolor(red) || hist f620,bin(30) blcolor(yellow) ||
function y=Fden(2,8,x), range(0 20) || function y=Fden(6,20,x), range(0 20)
*=====end=====

```



12.6 t 分布：未知总体方差时的抽样分布

t 分布是标准正态分布与卡方分布除自由度取平方根后的比值。当用样本方差代替总体方差后，它服从自由度为 $n-1$ 的 t 分布。

$$T = \frac{\bar{X} - \mu}{S/\sqrt{n}} \sim t(n-1) \quad (3)$$

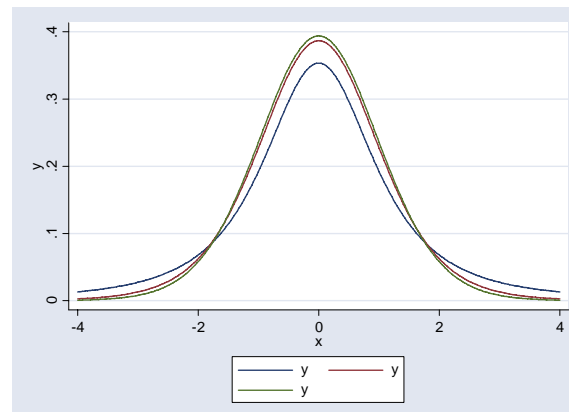
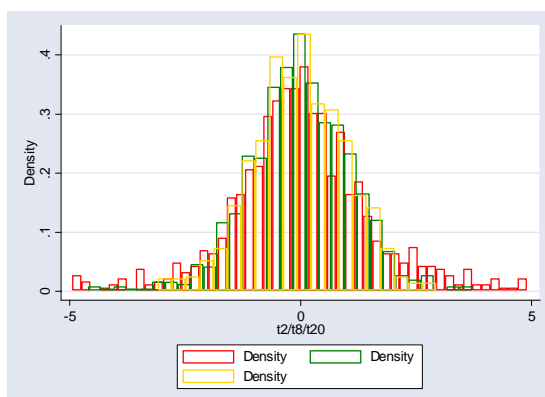
```

*=====begin=====
drawnorm x1-x20,n(1000) clear
forv i=1/20 {
  replace x`i'=x`i'^2
}
forv i=1/20 {
  egen y`i'=rsum(x1-x`i')
}
keep y1 y2 y4 y8 y20
drawnorm x,n(1000)
g t1=x/sqrt(y1)
g t2=x/sqrt(y2/2)
g t8=x/sqrt(y8/8)
g t20=x/sqrt(y20/20)
tw hist t2 ,bin(100) || hist t8 ,bin(100) || hist t20 ,bin(100)

```

```
tw function y= tden(2,x),range(-4 4) || function y= tden(8,x),range(-4 4) ||
function y= tden(20,x),range(-4 4)
```

```
*=====end=====
```



12.7 多元正态分布

```
*=====begin=====
```

```
clear
```

```
drawnorm x y, n(1000)
```

```
g z=exp(0.5*(-x^2-y^2))
```

```
scat3 x y z //在运行 sct3 之前需要先下载该命令, search scat3, net
```

```
scat3 x y z, msymbol(point) mcolor(gold) shadow(msize(0))
```

```
*=====end=====
```

```
clear
```

```
drawnorm x y, n(10000)
```

```
g z=exp(0.5*(-x^2-y^2))
```

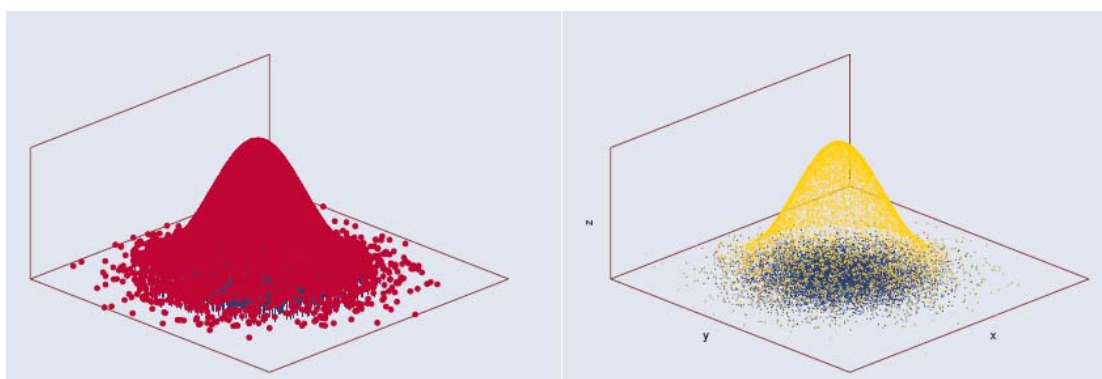
```
g y1=-x+invnormal(uniform())
```

```
g z1=exp(0.5*(-x^2-y1^2))
```

```
scat3 x y z, mcolor(gold) shadow(msize(0)) saving(1,replace)
```

```
scat3 x y1 z1, mcolor(gold) shadow(msize(0)) saving(2,replace)
```

```
graph combine 1.gph 2.gph
```



```
clear
drawnorm x y, n(10000)
g z=exp(0.5*(-x^2-y^2))
g y1=x^3+invnormal(uniform())
g z1=exp(0.5*(-x^2-y1^2))
scat3 x y z, mcolor(gold) shadow(msize(0)) saving(1,replace)
scat3 x y1 z1, mcolor(gold) shadow(msize(0)) saving(2,replace)
graph combine 1.gph 2.gph
```

13 参数估计与假设检验

13.1 极大似然估计的原理

极大似然的估计原理可以由下面的程序得到说明。我们首先生成 10 个服从正态分布的总体，每个总体的均值都不同，依次为 0, 1, 2, 3, 4, 5, 6, 7, 8, 9。方差相同，均为 1。然后我们随机地取出一个总体，从中抽出 10 个样本，因为事先不知道是从哪一个总体中抽出来的，所以我们分别用已知的 10 个总体参数值代入似然函数，计算出 10 个似然函数值，取其中最大的似然值，认为该样本是从相应的总体中取出的（从而联合概率密度也最大化）。然后我们让计算机告诉我们它是从第几个总体中取样的，并与我们的判断进行对比。

```

=====begin=====
capt prog drop mle
prog mle
/*生成10个均值不同、方差均为1的正态总体，每个总体取8个样本*/
drawnorm double x0-x9,n(8) m(0,1,2,3,4,5,6,7,8,9) clear
global i=int(10*uniform()) //设定一个随机数，用于随机取出一个总体
forv j=0/9 {
gen lnf`j'=-0.5*ln(2*_pi)*8-sum(0.5*(x$i-`j')^2) //对取出的总体计算似然值
scalar lnf`j'=lnf`j'[_N] //最终的似然值
}
scalar list // 比较10个似然值哪个最大，猜想是从第几个总体取出来的？
end
mle
*根据10个似然值，猜想是从第几个总体取出来的？
di "所抽中的样本为" as error "%i" $i //显示真正的取样总体是什么
=====end=====

```

在现实中，我们并不知道任何一个真正的总体参数，因此，只能借助于找到样本似然值（实际上是联合概率密度的对数值）最大的总体参数，即认为其是总体参数。在 STATA 中实现最大似然法的估计必须自己编写程序。下面的例子说明了如何利用 stata 编写程序来实现对模型的极大似然估计。

13.2 正态总体均值和方差的极大似然估计

```

=====begin=====
capt prog drop bb
prog bb //定义程序的名称
args lnf u v //声明参数,u 为均值, v 为方差
quietly replace `lnf' = -0.5*ln(2*_pi) - ln(v) -0.5*($ML_y1-`u')^2/(`v')^2
end
drawnorm x,n(100) m(10) sd(3) clear//模拟均值为10，方差为3的100个正态样本
ml model lf bb (x=) (variance:) //利用迭代法则进行极大似然估计

```

```
ml maximize //执行模型估计
```

```
*=====end=====
```

args 声明输入项，lnf 是一个暂元，用于存放每次迭代得到的似然函数值，u 和 v 表明我们的似然函数中有两个待估参数值，即均值和标准差。

quietly replace 定义单个似然函数形式，这里是正态分布的似然函数。replace 对每一个观察值求得似然函数值后，按观察值的顺序依次累加后存入暂元 `lnf' 中，所有观察值的似然函数值之和将保存在 `lnf'[_N] 中。程序中 \$ML_y1 是专门用于存放被解释变量的，它是 stata 指定的一个全局暂元。

ml model 是 stata 自带的定义最大似然函数估计“完整架构”的基本指令，bb 是我们上面列出的用于进行最大似然估计的程式的名称。被解释变量可以放在任何一个括号中，但一定要放在等号的左边，如果现行约束式中不需要输入解释变量，那么可以只写一个括号，里面不放任何东西。第二个括号中是对方差的估计。

ml maximiz 命令执行后我们会看到输出结果，估计出的总体均值和方差。

13.3 最小二乘估计 OLS 原理

与极大似然估计寻求样本密度函数对数值最大不同，最小二乘估计寻求样本点与总体参数的距离最小。这种距离通常以平方和来表示，因此称为最小二乘估计。最小二乘估计原理可以由下面的程序得到说明。

我们首先生成 10 个服从正态分布的总体，每个总体的均值都不同，依次为 0, 1, 2, 3, 4, 5, 6, 7, 8, 9。但方差相同，均为 1。然后我们随机地取出一个总体，从中抽出 10 个样本，因为不知道是从哪一个总体中抽出来的，所以我们分别计算样本点与已知的 10 个总体参数值（均值）之间距离的平方和，然后取使得平方和最小的哪个参数值，认为该样本是从哪个总体中取出的。然后我们让计算机告诉我们它是从第几个总体中取样的。

```
*=====begin=====
```

```
capt prog drop ols
```

```
prog ols
```

```
drawnorm double x0-x9,n(8) m(0,1,2,3,4,5,6,7,8,9) clear
```

```
*生成10个总体,每个总体取8个样本
```

```
global i=int(10*uniform()) //设定一个一位数的随机数, 0, 1, 2, ..., 9
```

```
forv j=0/9 {
```

```
gen lnf`j'=sum((x`i'-`j')^2) //对某总体计算10个观察值到总体均值的平方和
```

```
scalar lnf`j'=lnf`j'[_N] //取平方和赋给标量
```

```
}
```

```
scalar list //比较10个平方和哪个最大,猜想是从第几个总体取出来的?
```

```
end
```

```
ols
```

```
*根据10个平方和,猜想是从第几个总体取出来的?
```

```
di "所抽中的样本为" as error "X" $i //显示真正的取样总体是?
```

```
*=====end=====
```

13.4 矩估计 MM 原理

矩估计法不需要知道总体的分布，只要知道总体的矩即可。然后直接求出样本矩，代替总体矩即可。矩估计原理可以由下面的程序得到说明。

我们首先生成 10 个总体，每个总体的均值都不同，依次为 0, 1, 2, 3, 4, 5, 6, 7, 8, 9。然后我们随机地取出一个总体，从中抽出 8 个样本，因为不知道是从哪一个总体中抽出来的，所以我们分别计算样本的一阶原点矩，即均值，然后取该样本均值作为总体均值的估计值，比较样本均值与总体均值，认为相距最近的哪个样本是从相应总体中取出的。

```

=====begin=====
cap prog drop mm
drawnorm double x0-x9,n(8) m(0,1,2,3,4,5,6,7,8,9) clear
*生成10个总体,取样,得到10个样本
global i=int(10*uniform()) //设定一个一位数的随机数, 0, 1, 2, ..., 9
quietly sum x$i
di r(mean)
*根据样本均值,猜想是从第几个总体取出来的?
di "所抽中的样本为" as error "X" $i " //显示真正的取样总体是?
=====end=====

```

13.5 区间估计原理

区间估计与点估计不同，它寻求一个区间，该区间以一定的概率保证真正的总体参数值包含在其中，当然，对于一个特定的样本，它可能包含参数真值，也可能不包含。

```

=====begin=====
capt prog drop bb
prog bb
drawnorm x,n(100) m(5) sds(10) d clear
/*生成一个均值u=5,标准差o=10的正态随机变量样本,样本容量为100*/
quietly sum x
end
***将上述抽样试验进行100次,得到100个样本均值mean和标准差sd
simulate mean=r(mean) sd=r(sd), reps (100) nodots: bb
g n=_n
*在已知总体方差前提下(总体标准差为10),求100个子样本95%的置信区间
g zlow=mean-invnorm(0.975)*10/sqrt(100)
g zhigh=mean+invnorm(0.975)*10/sqrt(100)
*在总体方差未知的前提下,用样本标准差sd替代,需要借助t统计量
g tlow=mean-invttail(99,0.025)*sd/sqrt(100)
g thigh=mean+invttail(99,0.025)*sd/sqrt(100)
*考察总体均值是否在子样本的95%置信区间内,如不在则标记为1,否则为零
g zsign=(zlow<5& zhigh>5)

```

```
g tsign=(tlow<5& thigh>5)
```

*统计没有包括总体均值的子样本95%置信区间个数

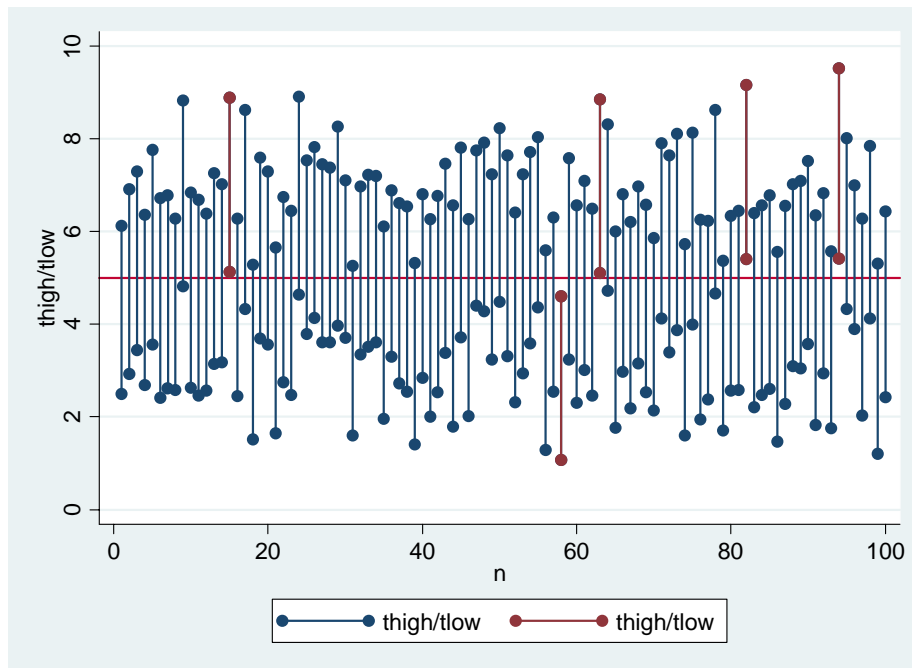
```
table zsign
```

```
table tsign
```

*图示

```
tw rcapsym thigh tlow n, yline(5) || rcapsym thigh tlow n if thigh<5 | tlow>5
```

```
*=====end=====
```



13.6 假设检验原理

由于（1）我们不知道真正的总体均值，（2）只能抽取一个样本，因此，当我们根据对样本的统计结果来对总体均值进行推断的时候就可能出错。从一次抽样来推断时，样本均值的 95% 置信区间可能含有总体均值，也可能不含有总体均值，含有的可能性为 95%，不含有的可能性为 5%。如果本来是不含有的真值的，我们认为含有，就犯了纳伪错误；如果本来是含有的真值的，我们认为不含有，就犯了弃真错误。

当总体的方差未知时，用样本方差来替代总体方差，则样本均值减总体均值服从(n-1)的 t 分布。

情形 1： 总体均值已知，为 $\mu=5$ 。但我们假装不知道，却做出了对总体均值正确的原假设，认为它等于 $\mu_0=5$ ，则抽样进行假设检验如下

```
*=====begin=====
```

```
drawnorm x,n(100) m(5) sds(10) d clear
```

*生成一个均值 $\mu=5$ ，标准差 $\sigma=10$ 的正态随机变量，作为研究总体

```
quietly sum x
```

```
di as text "从样本计算t统计值为: " as error (r(mean)-5)/(sqrt(100)*r(sd))
```

```
di as text "根据t统计量临界值为: " as error invttail(99,0.025)
```

```
di as result "对这次实验，拒绝还是接受?"
```



```
*=====end=====
```

由于我们通常只取一次样，所以有可能碰巧得到的样本正好是导致我们拒绝真的原假设的样本。这时我们就会犯错误。然而，弃真错误的可能性比较小。在100次这样的抽样研究中，大概有5次左右。

将上述试验进行100次，统计一下有多少次拒绝，多少次接受？

```
*=====begin=====
```

```

capt prog drop bb
prog bb
drawnorm x,n(100) m(5) sds(10) d clear
quietly sum x
scalar ref=(abs(sqrt(100)*(r(mean)-5)/r(sd))>invttail(99,0.025))
*如果样本统计量 ( t ) 值大于临界值，则拒绝原假设一次 jud=1, 否则为0
end
simulate ref, reps(100):bb
tab _sim                                //其中的1表示在100次中拒绝原假设的次数。

```

```
*=====begin=====
```

情形 2：总体均值已知，为 $\mu=5$ 。但我们假装不知道，并做出了对总体均值错误的原假设，如认为它等于 $\mu_0=10$ ，则抽样进行假设检验如下

```
*=====begin=====
```

```

capt prog drop bb
prog bb
drawnorm x,n(100) m(5) sds(10) d clear
quietly sum x
scalar ref=(abs(sqrt(100)*(r(mean)-10)/r(sd))>invttail(99,0.025) )
end
simulate jud, reps(100):bb
tab _sim

```

```
*=====begin=====
```

这时，我们 100 次地拒绝了原假设，认为原总体的均值不可能为 10。

14 简单回归原理

目标：理解简单回归的原理，能手工推导并计算出回归结果中的每一项。注意：分清哪些是现实可得的数据，哪些是理想模型。

14.1 回归分析原理

计量经济学家的根本任务就是从现实中挖掘变量之间的关系。

各种变量用X、Y等字母表示，而这些变量的关系往往体现为他们之间的函数关系 $Y=f(X)$ 。函数关系刻画了某个变量如何伴随着另一个（些）变量的变化而变化。

借用大家都熟悉的物理学例子，炮弹以初速度 v 和与水平轴成 θ 角离开地面，从发射点到落点的距离为 d ，根据牛顿定律可以得到

$$d = \frac{v^2}{g} \sin 2\theta \quad (1)$$

这似乎是一个不受限制的满足因果律的函数关系：固定发射角，给定一个初速度，就得到一个确定的距离。其实并非如此，它只是一个近似公式，是人类思维的发明，是一个理想化模型的解。这个理想化模型忽略了空气阻力、气压、重力加速度的变化以及初速度和发射角的不精确性。因此，它不是现实问题的解，只能在一定限制条件下应用，即如果所忽略的诸因素小于 δ ，则（1）式的误差小于 ϵ ，在可以接受的误差范围内，才可以放心地进行这种简化推测。

现在假设落点区域由许多小洞组成，我们想确定落入洞的位置。由于 v 和 θ 的不精确，以及炮弹在空气中受到的扰动，我们无法给出这一问题的确定解。但是在一次确定的发射中，能够发射完毕之后测定特定的炮弹会落入哪个洞，记为 d_i 。但是这个 d_i 可能与按（1）式计算的结果不一致，事前计算出的这个结果记为 \hat{d}_i 。换言之，单次实验的结果看似确定的，但这种确定是事后的，事前它是不确定的，是随机现象。因此，第 i 次实验的事后结果可表达为：

$$d_i = \frac{v_i^2}{g_i} \sin 2\theta_i + u_i$$

它的理想状态（近似状态：假设有精确的初速度 V 和精确的发射角 θ ，有精确的重力加速度 g ，因此可以事前确定落点），可以表达为

$$\hat{d}_i = \frac{v^2}{g} \sin 2\theta$$

如果假定 V 、 θ 和 g 是精确的，但炮弹在空中飞行时受到干扰，这种干扰产生的影响为 u_i 则

$$d_i = \hat{d}_i + u_i = \frac{v^2}{g} \sin 2\theta + u_i$$

上式中仅有 u_i 和 d_i 为随机变量。

如果我们提出另外一个问题，既然单次实验结果是随机的，是事前不能确定的，当我们用同一个大炮在同一地点打出许多有相同初速度和发射角的炮弹时（注意，这也只是理想化状态），这些炮弹落入第 j 个洞的概率是多少？则这一问题不再有因果答案，只能给出一个随机性的解释，表现为概率因果模式。在这种情况下，看似随机的结果却又表现出确定性，其确定性表现为：“如果试验次数足够多，平均特性具有高度的确定性”。

因此，同一个问题既可用确定的因果方式也可用概率的方式进行分析。有人可能会说，这本来就是确定性的问题，尽管我们不知道，但炮弹总有其精确初速度和发射角，如果我们知道它，就能准确知道它落入洞的位置，因此，我们之所以需要概率解释是因为我们对一些因素的无知。

对这一争议的回答是：科学家并不关心什么是真实的，只关心什么是他们能够观测到的。这也

是现代关于真实的观点。爱因斯坦说：“数学定律不能百分之百地确实地用在现实生活里，能百分之百地确实地用数学定律描述的，就不是现实生活。”，如果我们接受，事实上我们必须接受，科学理论不是自然规律的发现，而是人类思维的发明，那么，因果性和随机之间，或者确定性和或然性之间是没有抵触的。

现在，给定初速度 v_0 ，发射角 θ_0 ，在同样的条件下发射 n 次炮弹。每一次都会受到微小因素的干扰而产生误差 u_i 。这些误差将服从某种分布，设其分布的密度函数为 $f(u|v=v_0)$ ，因为

$$d_i = \frac{v_0^2}{g} \sin 2\theta_0 + u_i \quad (i=1,2,\dots,n)$$

则给定初速度 v_0 和发射角 θ_0 ，落点距离 d_i 服从以 v_0 和 θ_0 为条件的分布 $f(d|v_0, \theta_0)$ 。如右图

如果初速度改变为 v_k ，保持发射角不变，得到

$$d_j = \frac{v_k^2}{g} \sin 2\theta_0 + u_j \quad (k=1,2,\dots,m)$$

不断改变初速度，将得到若干 d 对应于 v 的分布如图。

如果我们已知初速度为 v_0 时随机误差的分布函数，比如它服从方差为 σ_0^2 ，均值为 u_0 的正态分布，则落点在 $v=v_0$ 时的均值就可以计算出来，它正好是 d 的误差条件均值 u_0 与事前理论计算结果之和，为一个确定的数（可以理解为初速为 θ_0 时无穷次发射的平均落点）。

$$E(d|v=v_0) = \frac{v_0^2}{g} \sin 2\theta + u_0$$

那么这些落点的方差是多少呢？

$$\text{var}(d|v=v_0) = \text{var}\left(\frac{v_0^2}{g} \sin 2\theta + u|v=v_0\right) = \text{var}(u|v=v_0) = \sigma_0^2$$

显然，条件方差也是由分布规律决定的，它也是一个确定的结果（给定初速度，无穷次实验将得到的一个稳定的值）

类似地可以求得其他初速度下的落点期望值和方差，

$$E(d|v=v_k) = \frac{v_k^2}{g} \sin 2\theta + u_k$$

$$\text{var}(d|v=v_k) = \text{var}\left(\frac{v_k^2}{g} \sin 2\theta + u|v=v_k\right) = \text{var}(u|v=v_k) = \sigma_k^2$$

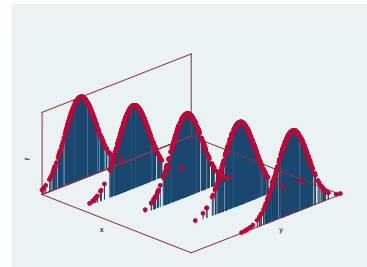
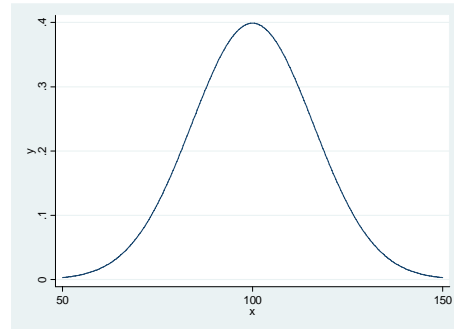
可见条件期望值和条件方差均依赖于初速度。

如果已知条件分布（包括分布类型和分布参数），则相应的条件期望和条件方差可以直接计算得到。比如在发射角为45度时，当初速度为100时，误差服从正态分布，而且假设 $u_0=4$ ， $\sigma_0=0.09$

$$E(d|v=1) = \frac{v_0^2}{g} \sin 2\theta + u_0 = \frac{10000}{9.8} + 4$$

$$\text{var}(d|v=1) = \sigma_0^2 = 0.09$$

这是理论推导的结果，要求他能自圆其说，还要求它能与观察一致，于是我们抽取一个样本（进行有限次试验），分析落点距离，如果它服从正态分布 $N(10000/9.8+4, 0.09)$ ，则观察将与理论一致。我们可以放心地使用这个结果，并且能根据目标的大小，在一定的把握下（95%），确定需要几发炮弹可以炸中目标。



然而，在未知世界面前，在黑箱没有被伟大的牛顿打开之前，我们并不知道下面的这个式子

$$d = g(v) = \frac{v^2}{g} \sin 2\theta$$

也即在牛顿之前，这是一个未知的理论，是我们要探索的一个表达式。即

$$d = g(v) = ?$$

现在，我们假装我们不知道这个理论，不知道 $g(v)$ 的复杂函数形式。由于 $g(v_k)$ 为抽象的未知的函数形式，因此，借助于泰勒展开式，得到

$$d = g(v) = \beta_0 + \beta_1 v + \beta_2 v^2 + \dots + o(v)$$

如果仅取一次项，有

$$d = g(v) = \beta_0 + \beta_1 v + o(v)$$

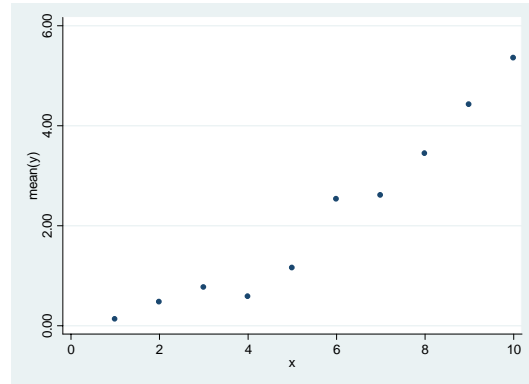
考察发射炮弹的情况，当初速度为 v_k 时，第 i 次发射事后观察到的落点可表达为：

$$d_i = \beta_0 + \beta_1 v_k + o(v_k) + u_i$$

如果以初速度 v_k 多次发射，其落点期望值为

$$E(d | v_k) = E(\beta_0 + \beta_1 v_k + o(v_k) + u_i | v_k) = \beta_0 + \beta_1 v_k + o(v_k) + u_k \quad (2)$$

当条件分布 $f(u | v_k)$ 已知时， u_k 为确定的常数。而当 v_k 变化时， u_k 也变动，但它是相对于 v_k 的常数。在控制实验中， v_k 是给定的数，因此，当 $f(u | v_k)$ 已知时，在 (2) 式右端，仍然未知的还有高阶项（由于不知其具体形式）和参数 β_0 和 β_1 的具体值。如果可以忽略并且忽略掉高阶项（假设正确设定函数形式，即 $o(v_k)=0$ ），并用某种办法确定出 $\beta_0 = \hat{\beta}_0$ 和 $\beta_1 = \hat{\beta}_1$ ，我们即可以得到一个近似的关于 $g(x)$ 的关系式，并完成探索任务。 $d = g(v) = \beta_0 + \beta_1 v = \hat{\beta}_0 + \hat{\beta}_1 v$ ，那么又如何确定 $\beta_0 = \hat{\beta}_0$ 和 $\beta_1 = \hat{\beta}_1$ 呢？



对于任意一个 v_k ，设 $v_k=5$ 百米/秒，经过无穷次实验，可以“观察到” $E(d | v_k) = 200$ 百米，于是由 $E(d | v_k) = \beta_0 + \beta_1 v_k + u_k$ 可以得到

$$200 = \beta_0 + 5\beta_1 + u_k \quad (3)$$

显然，当自变量 v 没有任何变异时，无法获得 β_0 和 β_1 的具体解。

改变初速度为 $v_m=8$ 百米/秒，再进行无穷次实验，可以“观察到” $E(d | v_m) = 300$ 百米，于是，

$$300 = \beta_0 + 8\beta_1 + u_m \quad (4)$$

(4) 式减 (3) 式，就得到

$$100 = 3\beta_1 + u_m - u_k \quad (5)$$

当所有的条件分布已知时，比如 $u_k=2$ ， $u_m=3$ ，则 $\beta_1 = [100 - (3-2)]/3 = 33$

然而，现实中，这些条件分布不可能是已知的，它也是极难确定的未知世界。这样，(5) 式仍然有三个未知参数，无法确定出 β_1 的具体值。但是我们注意到，如果有 $u_k = u_m$ ，即使我们不知道 u_k 和 u_m 的具体值，(5) 式也会被简化成

$$100 = 3\beta_1$$

并能够很容易获得其答案。

因为 u_k 和 u_m 为条件均值，他们相等也即 $E(u | v_k) = E(u | v_m)$

$$\text{误差的条件期望相等 } E(u | v) = u^* \quad (6)$$

误差的条件均值相等意味着什么呢？如果 (6) 式成立，则下面的 (7) 式和 (8) 式也成立（注意到 u^* 为常数）。

$$E(u) = E[E(u|v)] = u^* \\ \text{由 } E\tilde{u} = E(u - u^*) = u^* - u^* = 0 \quad (7)$$

$$\text{cov}(v, \tilde{u}) = \text{cov}(v, u) = E[(v - Ev)(\tilde{u} - E\tilde{u})] = E\tilde{u}v = 0 \quad (8)$$

而

$$d_i = \beta_0 + \beta_1 v_k + u_i \\ d_i = (\beta_0 + u^*) + \beta_1 v_k + u_i - u^* = \beta_0^* + \beta_1 v_k + \tilde{u}_i \\ E(d|v_k) = \beta_0^* + \beta_1 v_k + E(\tilde{u}_i|v_k) = \beta_0^* + \beta_1 v_k \quad (9)$$

上式右端仅随 v 而变化，其余的两个 β 均为未知常参数，只要确定这两个常参数即可。如果取 v_k 的两个值，分别做无数次实验得到两个距离的条件均值，代入（9）式，解一元二次方程即可得到 β 值。但是要注意到这里的 β_0^* 不是 β_0 ，它有一个 u^* 的偏误。

再强调一下获得（9）式的若干假设条件：

SLR1：自变量不能唯一，至少要两个不同取值；

SLR2：函数形式正确设定，高阶项为零；

SLR3：初速度改变后各误差的条件均值相等；

我们希望通过观察（实验）来确定出两个常参数，并获得这个函数。于是，我们固定发射角做实验，通过不断改变初速度（改变10次， $x=1-10$ ），得到10000个数据（模拟数据bomb²）。对每个 x 求 y 的均值 $E(d|v)$ ，得到下表。

v	1	2	3	4	5	6	7	8	9	10
$E(d v)$.13	.47	.76	.58	1.15	2.53	2.61	3.44	4.43	5.35

如何估计呢？可能会有人想到采用下面的方程式，

$$E(d|v=1) = \beta_0^* + \beta_1 \times 1 = 0.13$$

$$E(d|v=2) = \beta_0^* + \beta_1 \times 2 = 0.47$$

...

$$E(d|v=10) = \beta_0^* + \beta_1 \times 10 = 5.35$$

显然，上面的式子互相矛盾，无法得到 β 的估计。为什么会这样呢？

因为无论我们实验次数 n 有多大，只要是有限的，则它就与真正的总体参数 $E(d|v)$ 就有一定距离，也就一定会存在抽样误差 ε ，我们计算得到的落点距离 d 的各个条件均值实际上只是对真正的 d 的条件均值的一个估计值，设该估计值离真正的条件均值的距离为 ε 。则正确的表达式为

$$E(d|v=1) = E(d|v=1) + \varepsilon_1 = n_1^{-1} \sum_{i=1}^{n_1} d_i + \varepsilon_1 = 0.13 + \varepsilon_1$$

如果上述三个关键假设满足，则

$$E(d|v=1) = \beta_0^* + \beta_1 \times 1 = \bar{d}_1 + \varepsilon = 0.13 + \varepsilon_{11}$$

对任意一个初速度，有

$$\beta_0^* + \beta_1 \times v_k = \bar{d}_k + \varepsilon_k$$

```

2 clear
set obs 10000
gen x=int((_n-0.5)/1000)+1
gen u=9*invnorm(uniform())
gen y=x^2/9.8*sin(_pi/6)+u
table x, c(mean y) format(%5.2f)

```

将k个式子取平均，得到

$$\beta_0^* + \beta_1 \times \bar{v} = \bar{d} + \bar{\varepsilon}$$

两式相减，有

$$\beta_1 \times (v_k - \bar{v}) = (\bar{d}_k - \bar{d}) + (\varepsilon_k - \bar{\varepsilon})$$

仍然无法给出 β 的估计值，因为方程的个数小于未知数的个数。

我们来考虑其他的途径。从下式出发

$$d_i = \beta_0 + \beta_1 v_i + u_i = \beta_0^* + \beta_1 v_i + \tilde{u}_i$$

前面已论证过，为得到上式并估计出 β ，（7）式和（8）式必须成立，即

$$E\tilde{u} = 0 \quad (7)$$

$$E\tilde{u}v = 0 \quad (8)$$

这两个条件实际上是随机项要满足的两个矩条件，称为总体矩条件。如果我们从总体中抽取样本，而且SLR4：样本是独立随机抽取的，那么样本将传承总体的性质，总体要满足的条件，样本也将近似地得到满足。（7）式和（8）可以表达为

$$E\tilde{u} = E(d_i - \beta_0^* - \beta_1 v_i) = 0$$

$$E\tilde{u}v = E[\tilde{u}(d_i - \beta_0^* - \beta_1 v_i)] = 0$$

抽取随机样本，计算相应的样本矩，令样本矩等于总体矩，实际上即相应样本矩等于零

$$\hat{E}\tilde{u} = \frac{1}{n} \sum_{i=1}^n (d_i - \beta_0^* - \beta_1 v_i) = 0$$

$$\hat{E}(\tilde{u}v) = \frac{1}{n} \sum_{i=1}^n v_i (d_i - \beta_0^* - \beta_1 v_i) = 0$$

由上述两式，进行代入换算，即可以得到：

$$\hat{\beta} = \frac{\sum (v_i - \bar{v})(d_i - \bar{d})}{\sum (v_i - \bar{v})^2}$$

这一结果是无偏的，其证明过程参见课本P46页。

14.2 模拟实验

下面，我们来做进一步的模拟实验。首先，作为一个基准，考虑一种最理想状态，当发射角为45度，初速度取（1，10）之间的正整数，没有测量误差，重力加速度恒定，空气阻力等炮弹飞行中的扰动产生的条件误差服从均值为零，方差为0.09的正态分布，于是可以根据牛顿定律，可生成一个总体如下。

$$d_i = \frac{v_i^2}{g} \sin 2\theta + u_i = \frac{1}{9.8} v_i^2 + u_i = 0.102 v_i^2 + u_i \quad u_i \sim N(0, 0.09)$$

*****函数形式设定问题*****

```
*****begin*****
capt prog drop bomb
prog bomb
scalar x=1+int(10*uniform()) //给定一个在（1，10）中取正整数的随机初速度
scalar y=x^2/9.8*sin(_pi/2)+0.09*invnorm(uniform()) //随机落点距离
end
clear
```

```
simulate "bomb" x=x y=y, reps(30) //随机独立试验30次，得到30个样本。
```

```
reg y x // 简单回归y=a+bx+u，注意到回归系数的值
di 1/9.8*sin(_pi/2) //总体参数值在95%的置信区间范围内吗？
```

```
gen x2=x^2 // 正确的函数形式设定，y=a+bx2+u
reg y x2 // 简单回归，注意系数估计值
di 1/9.8*sin(_pi/2) //总体参数值在95%的置信区间范围内吗？
```

```
*=====end=====
```

分析上面的程序结果，我们发现，**当函数形式误设时**，如本来**应该设定**X为平方项，却只取了一次项，将得到有偏的估计。在正确设定函数形式的情形下，得到无偏的估计结果，表现为真正的总体参数值0.102在95的置信区间内。

第二个实例：函数形式误设中的遗漏变量偏差问题：

```
*=====end=====
```

```
clear
mat m=(3,4,5,0)
mat sd=(9,9.6,0.01,0\9.6,16,12,0\0.01,12,25,0\0,0,0,1)
/*生成假设的数据集,该数据集中X1(教育)与X2(能力)高度相关,教育与X3(经验)不相关,
能力与经验相关,所有变量与相貌都不相关*/
capt prog drop bb
prog bb
drop _all
drawnorm x1 x2 x3 x4,n(30) means(m) cov(sd)
gen y=120+50*x1+100*x2+30*x3+x4
quietly reg y x1 x2 x3
end
simulate _b _se, reps(1000): bb
hist _b_x1
```

14.3 回归报告结果中各项的手工计算

在上面的推导过程中，我们得到过下式

$$\text{var}(d | v = v_k) = \text{var}\left(\frac{v_0^2}{g} \sin 2\theta_0 + u | v = v_k\right) = \text{var}(u | v = v_k) = \sigma_k^2$$

假设所有的条件方差均相等（SLR5：同方差假定），则有

$$\text{var}(d | v) = \text{var}(u | v) = \sigma_k^2 = \sigma^2$$

根据p662的公式，有

$$\text{var}(d | v) = \text{var}(u | v) = E(u^2 | v) - [E(u | v)]^2 = \sigma^2$$

因为条件均值为零（或常数），所以有

$$E(u^2 | v) = \sigma^2$$

在正确函数形式设定假设下，有

$$E(u^2 | v) = E[(d_i - \beta_0 - \beta_1 v_k)^2 | v] = \sigma^2$$

如果 u 与 v 独立, 有p662

$$E(u^2) = E(u^2 | v) = \sigma^2$$

因为残差满足下式

$$\hat{u}_i = d_i - \hat{\beta}_0 - \hat{\beta}_1 v_i = \beta_0 - \beta_1 v_i + u_i - \hat{\beta}_0 - \hat{\beta}_1 v_i = u_i - (\hat{\beta}_0 - \beta_0) - (\hat{\beta}_1 - \beta_1) v_i$$

将上式取平均, 由于残差的平均值为零,

$$0 = \bar{u}_i - (\hat{\beta}_0 - \beta_0) - (\hat{\beta}_1 - \beta_1) \bar{v}_i$$

两式相减, 得到

$$\hat{u}_i = u_i - \bar{u}_i - (\hat{\beta}_1 - \beta_1)(v_i - \bar{v}_i)$$

将 d 写为 y , v 写为 x , 运算过程如下:

- (1) $\hat{u}_i = y_i - \hat{\beta}_0 - \hat{\beta}_1 x_i$
 $= (\beta_0 + \beta_1 x_i + u_i) - \hat{\beta}_0 - \hat{\beta}_1 x_i$
 $= u_i - (\hat{\beta}_0 - \beta_0) - (\hat{\beta}_1 - \beta_1) x_i$
- (2) $\bar{\hat{u}}_i = \bar{u} - (\hat{\beta}_0 - \beta_0) - (\hat{\beta}_1 - \beta_1) \bar{x}$
- (3) $\hat{u}_i = (u_i - \bar{u}) - (\hat{\beta}_1 - \beta_1)(x_i - \bar{x})$
- (4) $\hat{u}_i^2 = (u_i - \bar{u})^2 + (\hat{\beta}_1 - \beta_1)^2 (x_i - \bar{x})^2 - 2(\hat{\beta}_1 - \beta_1)(u_i - \bar{u})(x_i - \bar{x})$
- (5) $\sum \hat{u}_i^2 = \sum (u_i - \bar{u})^2 + (\hat{\beta}_1 - \beta_1)^2 \sum (x_i - \bar{x})^2 - 2(\hat{\beta}_1 - \beta_1) \sum u_i (x_i - \bar{x})$
- (6) $E(\sum \hat{u}_i^2) = \sum E[(u_i - \bar{u})^2] + E[(\hat{\beta}_1 - \beta_1)^2] \sum (x_i - \bar{x})^2 - 2E[(\hat{\beta}_1 - \beta_1) \sum u_i (x_i - \bar{x})]$
 $= (n-1)\sigma^2 + \sigma^2 - 2\sigma^2 = (n-2)\sigma^2$
- (7) $E(\frac{1}{n-2} \sum \hat{u}_i^2) = \sigma^2$, 故: $\hat{\sigma}^2 = \frac{1}{n-2} \sum \hat{u}_i^2$ 是 σ^2 的无偏估计量

下面的程序旨在了解regress命令, 首先我们手工计算出各项, 然后来考察regress命令

```

=====begin=====
cap prog drop bomb
prog bomb
  scalar x=1+int(10*uniform())
  scalar y=x^2/9.8*sin(_pi/2)+0.09*invnorm(uniform())
end
clear
simulate "bomb" x=x y=y, reps(30) //生成数据
gen x2=x^2 // 正确的函数形式设定, y=a+bx2+u
*  $\hat{\beta}_1 = \frac{\sum (x_i - \bar{x})(y_i - \bar{y})}{\sum (x_i - \bar{x})^2}$ 
*根据上式, 我们可以手工计算出b1的系数估计值
egen my=mean(y) //计算y的均值
egen mx=mean(x2) //计算x2的均值
gen nmy=y-my //中心化后的y
gen nmxx=x2-mx //中心化后的x2
egen fz=sum(nmxx*nmy) //上式的分子
egen fm=sum(nmxx^2) //上式的分母
egen sst=sum(nmy^2)

```



```

scalar b1=fz[_N]/fm[_N]           //系数b1的估计值
scalar b0=my[_N]-b1*mx[_N]       //系数b0的估计值

*  $\hat{\sigma}^2 = \frac{1}{n-2} \sum \hat{u}_i^2$  是  $\sigma^2$  的无偏估计量,根据该式可以计算回归方程的标准误估计值

gen uh=y-b0-b1*x2               //该命令得到残差
gen uh2=uh^2
sum uh2
scalar SSR=r(sum)               //计算出 SSR
scalar SST=sst[_N]             //计算 SST
scalar SSE=SST-SSR              // SSE
scalar R2=SSE/SST               //R2
scalar F=SSE/(SSR/(_N-2))       //F值
scalar PF=Ftail(1,_N-2,F)      //F值对应的P值
scalar SER=sqrt(SSR/(_N-2))     //误差标准差  $\hat{\sigma}$  (回归的标准误) 的估计结果

*  $Var(\hat{\beta}_1) = \frac{\sigma^2}{s_x^2} = \frac{\sigma^2}{\sum_{i=1}^n (x_i - \bar{x})^2}$ , 根据该式得到b1的标准误

sum x2
scalar SX2= r(Var)* (_N-1)       //这个结果为Sx2
scalar seb1= SER/sqrt(SX2)       //这个结果即是b1的标准误

scalar tb1= b1/seb1             //这个结果即是b1的t值
scalar pb1=ttail(_N-2,tb1)      //这个结果为b1的相应分位数p值。
scalar condn=b1-invttail(_N-2,0.025)*seb1 //95%置信区间的下限
scalar conup=b1+invttail(_N-2,0.025)*seb1 //95%置信区间的上限

reg y x2                       // 简单回归, 注意系数估计值
ereturn list                   //显示出回归结果存放的宏名及结果
di e(rss)                      // SSR被置于e(rss) 中
di e(rss)/(_N-2)               //这个结果即是对误差方差的估计结果, 请在回归结果中找到它
di e(rmse)                     //这个结果即是回归的标准误
sum x2
di _b[x2]/e(rmse)/sqrt(r(Var)*(_N-1)) //这个结果即是b1的t值, 请在回归结果中找到它
di ttail(_N-2, _b[x2]/e(rmse)/sqrt(r(Var)* (_N-1))) //这个结果为p值。
di _b[x2]+invttail(_N-2,0.025)*e(rmse)/sqrt(r(Var)* (_N-1)) //95%置信区间的上限
di e(mss)/(e(rss)/(_N-2))      //F值
di Ftail(1,28,e(mss)/(e(rss)/(_N-2))) //F值对应的P值

*=====end=====

```

14.3 线性模型的最大似然估计

$$e \sim N(0, \sigma^2)$$

$$y = \beta_0 + \beta_1 x + e \sim N(\beta_0 + \beta_1 x, \sigma^2)$$

由上式可知, 对于线性模型而言, 当 e 服从均值为零的正态分布时, y 服从均值为

$\beta_0 + \beta_1 x$ 的正态分布。这样，我们可以估计出该均值，不过此时该均值依赖于 x ，为条件均值。但是 β_0 与 β_1 是不变的总体参数。

用于估计的最大似然程式为：

```

=====begin=====
capt prog drop bb
prog bb
args lnf theta1 theta2
quietly replace `lnf' = ///
-0.5*ln(2*_pi)-ln(`theta2')-0.5*($ML_y1-`theta1')^2/(`theta2')^2
end

drawnorm x u,n(30) clear
gen y=10+8*x+invnorm(uniform()))
ml model lf bb (y= x) (variance:)
ml maximize
=====end=====

```

15 异方差模拟

目标：理解异方差的基本原理，知道异方差导致的后果，能检验异方差，能在出现异方差时，寻找合适的纠正办法。

15.1 条件分布图示

$y=x+e$ ，其中 $e \sim N(0, \sigma^2)$ ，因此 $y \sim N(x, \sigma^2)$ 。

同方差，条件均值满足函数关系 $E(y|x) = x$

clear

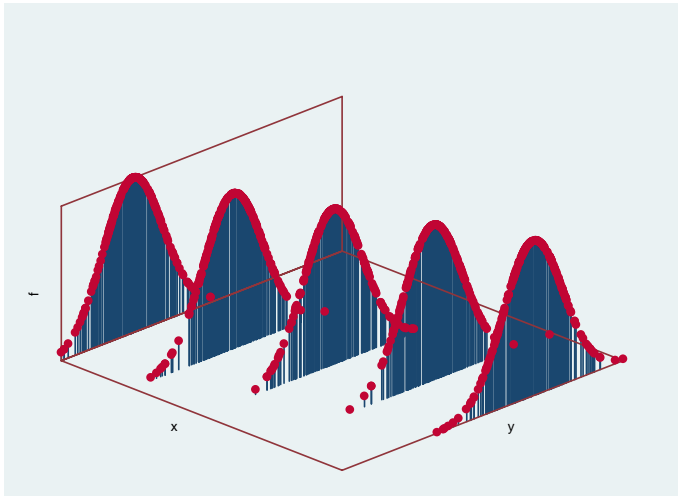
set obs 1000

gen x=int(5*uniform())+1 //生成自变量x，取值从1到5

gen y=x+invnorm(uniform()) //生成因变量y，取值为x+e

gen f=normden(y,x,1) //y的条件分布密度函数值

scat3 y x f //绘出三维图，scat3需要下载



$y=x+e$ ，其中 $e \sim N(0, \sigma_i^2)$ ，因此 $y \sim N(x, \sigma_i^2)$

异方差，条件均值满足函数关系 $E(y|x) = x$ ， $\text{var}(y|x_i) = \sigma_i^2$

clear

set obs 1000

gen x=int(5*uniform())+1

gen y=.

gen f=.

forvalue i=1/5 {

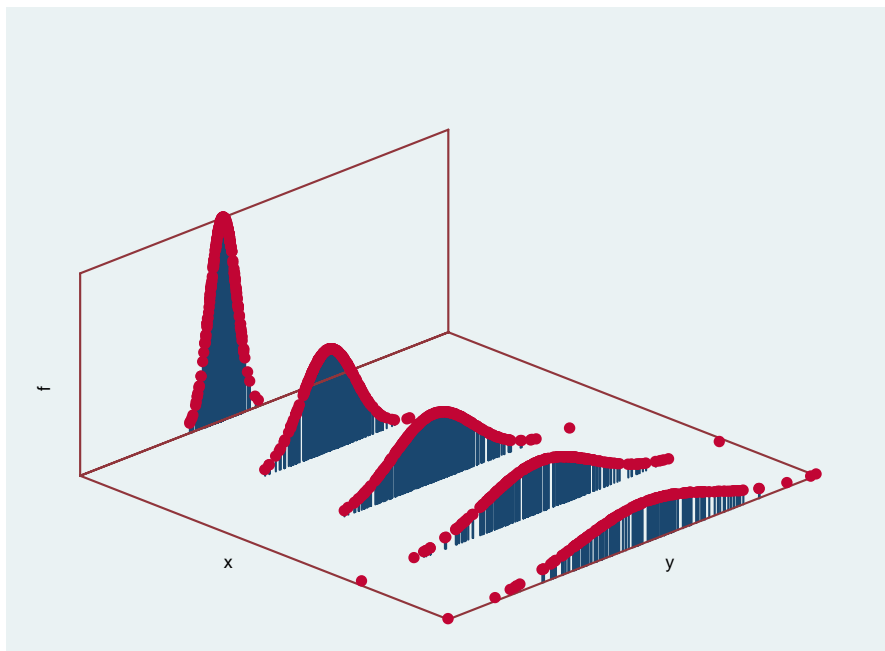
drawnorm e`i',n(1000) m(0) sds(`i') //误差项均值为零，方差为1到5

replace y=x+e`i' if x==`i'

replace f=normden(y,x,`i') if x==`i'

}

scat3 y x f



15.2 异方差的后果

异方差： u 的方差随着 x 的取值不同而不同。

❖ 同方差假定：MRL.5: $\text{Var}(u|x_1, \dots, x_k) = \sigma^2$

❖ 异方差(heteroskedasticity): $\text{Var}(u_i|x_{i1}, \dots, x_{ik}) \neq \sigma_i^2$

$$\hat{\beta}_{OLS} = \frac{\sigma^2}{SST_x}$$

(1) 忽视异方差性的 OLS 估计

- ❖ 该公式给出的方差是一个有偏误的估计量, 可能高估也可能低估.
- ❖ 忽视异方差而使用惯常的检验程序, 则无论得出什么结论或做出什么推断, 都可能产生严重的误导.

(2) 考虑异方差性的 OLS 估计

❖ 假定 σ 已知, 是否可以按该公式进行推断?

❖ 否! 因为 $\text{var}(\beta_{GLS}) < \text{var}(\beta_{ROLS})$

❖ 根据后者做出的置信区间将无谓地过大, 相应的 t 检验和 F 检验将可能提供不准确的结果, 使本来显著的系数变成统计不显著的.

❖ 但在大样本条件下, 稳健的异方差是一致估计量

$$\hat{\beta}_{ROLS} = \frac{\sum (x_i - \bar{x})^2 \cdot \sigma_i^2}{SST_x^2}$$

(3) 广义最小二乘估计 (GLS) 是 BLUE 的

❖ GLS 实质上是对满足标准最小二乘假定的转换变量的 OLS。其转化思路如下面的公式。

❖ 异方差中讨论的 GLS 和 WLS 可互换, 但 WLS 是更为一般的估计方法 GLS 的一种特殊情形。

$$y_i = \beta_0 + \beta_1 x_i + u_i \quad E(u_i^2) = \sigma_i^2$$

$$\frac{y_i}{\sigma_i} = \beta_0 \frac{1}{\sigma_i} + \beta_1 \frac{x_i}{\sigma_i} + \frac{u_i}{\sigma_i}$$

$$\text{var}\left(\frac{u_i}{\sigma_i}\right) = E\left(\frac{u_i}{\sigma_i}\right)^2 = \frac{1}{\sigma_i^2} E(u_i^2) = \frac{\sigma_i^2}{\sigma_i^2} = 1$$

$$\min \sum \left(\frac{\hat{u}_i}{\sigma_i}\right)^2 = \min \sum \left(\frac{y_i}{\sigma_i} - \beta_0 \frac{1}{\sigma_i} - \beta_1 \frac{x_i}{\sigma_i}\right)^2$$

$$= \min \sum (y_i - \beta_0 - \beta_1 x_i)^2 \frac{1}{\sigma_i^2}$$

戴维森和金农所做的蒙特卡罗实验：考虑下面的模拟

- ❖ $y = 5 + 8x + u_i$
- ❖ 上式中的 $u_i \sim N(0, x_i^a)$, 分别取 $a=0, a=1, a=2, a=3, a=4$ 模拟出一组数据, 利用这组数据分别采用 OLS, 稳健的 OLS 和 GLS 对斜率系数进行估计, 比较这些斜率系数的标准误。

```

=====begin=====
clear
capt prog drop het
prog het
drop _all
set obs 100
gen x=uniform()
gen u=sqrt(x^1)*invnorm(uniform()) //取误差项的形式为  $u_i \sim N(0, x_i^a)$ 
gen y=5+8*x+u
end

forvalue i=0/4 { //分别取a=0, a=1, a=2, a=3, a=4进行比较
quietly {
het `i' //当i=0时, 得到  $u_i \sim N(0, 1)$ ; 当i=4时, 得到  $u_i \sim N(0, x_i^4)$ 
reg y x //标准回归, 忽视异方差性的OLS估计
esti store ols`i' //将估计结果存入ols`i'
reg y x, robust //稳健回归
esti store rols`i'
gen w`i'=1/(x^`i') //获得权重
reg y x [aw=w`i'] //WLS估计
esti store wls`i'
replace y=y/sqrt(x^`i') //得到新的  $y^*=y/\sigma$ 
gen x0=1/sqrt(x^`i')
replace x=x/sqrt(x^`i') //得到新的  $x^*=x/\sigma$ 
reg y x0 x,noc // GLS估计,  $y^*=ax_0+bx_1^*+u$ , noc表示无常数项
esti stor gls`i'
}
esti table ols`i' rols`i' gls`i' wls`i',se //注意比较各种估计方法的标准误的大小
}
=====end=====

```

结论:不管是否考虑对异方差的修正, OLS 一致地过高估计了由(正确的)GLS 程序得到的真实标准误, 尤其是以大的 α 值为甚.

GLS 与经过变换后的最小二乘回归, 得到的估计量是完全一样的。

15.3 图形检验与怀特检验

简单情形

对如下的线性回归模型

$$y_i = \beta_0 + \beta_1 x_i + e_i$$

$$e_i \sim N(0, \sigma_i^2)$$

$$(i) \sigma_i^2 = \sigma^2$$

$$(ii) \sigma_i^2 = \alpha x_i^2$$

$$(iii) \sigma_i^2 = \alpha x_i$$

显然 (i) 不存在异方差, 而 (ii) 和 (iii) 存在异方差。模拟出数据, 然后分别用图形和怀特检验看是否能正确检验出异方差。

```

*=====begin=====
clear
set obs 1000
gen x=uniform()
gen u1=invnorm(uniform()) //同方差的误差结构
gen u2=x^2*invnorm(uniform()) //异方差的误差结构,  $u_{2i} \sim N(0, x_i^2)$ 
gen u3=x*invnorm(uniform()) //异方差的误差结构,  $u_{3i} \sim N(0, x_i)$ 
gen y1=1+5*x+u1
gen y2=1+5*x+u2
gen y3=1+5*x+u3
reg y1 x
rvpplot x //残差图, 以残差为Y轴, 以Y的拟合值 $\hat{Y}$ 为X轴
imtest,white //怀特检验, 零假设为同方差
reg y2 x
rvpplot x //残差图, 以残差为Y轴, 以x为X轴
imtest,white //怀特检验
reg y3 x
rvpplot x
imtest,white
*=====end=====

```

多元情形

二元线性回归模型

$$y_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + e_i$$

$$e_i \sim N(0, \sigma_i^2)$$

$$\sigma_i^2 = e^{\alpha_0 + \alpha_1 \ln(x_{i1}) + \alpha_2 \ln(x_{i2})}$$

$$(i) \alpha_1 = \alpha_2 = 0$$

$$(ii) \alpha_1 = 2, \alpha_2 = 0$$

$$(iii) \alpha_1 = 0, \alpha_2 = 2$$

$$(iv) \alpha_1 = \alpha_2 = 1$$

显然 (i) 不存在异方差，而 ii-iv 存在异方差。模拟出数据，然后分别用图形和怀特检验看是否能正确检验出异方差。

```

*=====begin=====
clear
set obs 1000
gen x1=uniform()
gen x2=1+3*invnorm(uniform())
gen e=invnorm(uniform()) //同方差情形
gen u2=exp(2*ln(x1))*e //异方差a1=2, a2=0, 只与x1有关
gen u3=exp(2*ln(x2))*e //异方差a1=0, a2=2, 只与x2有关
gen u4=exp(ln(x1)+ln(x2))*e //异方差a1=1, a2=1, 与x1和x2均有关
gen y1=1+3*x1+5*x2+e
gen y2=1+3*x1+5*x2+u2
gen y3=1+3*x1+5*x2+u3
gen y4=1+3*x1+5*x2+u4

reg y1 x1 x2
rvfplot

reg y2 x1 x2
rvpplot x1

reg y3 x1 x2
rvpplot x2

reg y4 x1 x2
rvfplot
imtest,white

```

小游戏：相邻两位同学，一位同学模拟出一套数据，记下模拟办法（可能存在异方差，也可能不存在）但不告诉另外的同学，仅将数据交给另外的同学，请其检验是否存在异方差，并猜测可能的异方差类型。然后交换进行游戏。

15.4 检验的功效(选读内容)

对如下的线性回归模型

$$y_i = \beta_0 + \beta_1 x_i + e_i$$

$$e_i \sim N(0, \sigma_i^2)$$

$$(i) \sigma_i^2 = \sigma_i$$

$$(ii) \sigma_i^2 = \sigma_i x_i^2$$

$$(ii) \sigma_i^2 = \sigma_i x_i$$

选择样本容量 $n=10, 20, 30, 40, 50, 60, 70, 80$ 分别在5%和10%的统计显著性水平下进行white检验，记录1000次模拟检验中拒绝零假设的百分比。

模拟过程如下

```

=====begin=====
clear
set more off
capt prog drop het
prog het
quietly {
drop _all
set obs `1' //`1'设置样本容量
gen x=uniform()
gen u1=invnorm(uniform()) //同方差情形
gen u2=x^2*invnorm(uniform()) //异方差的误差结构,  $u_{2i} \sim N(0, x_i^2)$ 
gen u3=x*invnorm(uniform()) //异方差的误差结构,  $u_{2i} \sim N(0, x_i)$ 
gen y1=1+5*x+u1
gen y2=1+5*x+u2
gen y3=1+5*x+u3
reg y1 x
imtest,white //异方差检验
scalar z1=r(p) //检验后的显著性P值保存到z1
reg y2 x
imtest,white
scalar z2=r(p)
reg y3 x
imtest,white
scalar z3=r(p)
}
end

forvalues i=10(10)80 { //当样本量为10, 20, ..., 70, 80时的检验功效
simulate z1 z2 z3,rep(100) nodots:het `i'
mkmat _s*,mat(A) //将检验的显著性P值存入矩阵A
mat B=(nullmat(B),A) //将不同样本下的检验的显著性P值存入矩阵B
}

svmat B //将矩阵B转换成数据集
drop _sim*
quietly forvalue i=1/24 {
gen a`i'=(B`i'>0.05) //将P值与0.05进行对比，大于则为1，否则为0
}

```



```
}
sum a*,seps(3)
```

***或者**

```
mata
A=st_matrix("B")
A=A:>0.05
(mean(A))'
end
```

***=====end=====**

注意到每三个为一组，第一个同方差，后两个异方差，在100次检验中，有多少次做出了正确的选择？

由于每一组的样本容量不同，注意随着样本容量的增大，检验的功效有什么变化？

下面的例题思路同上一个实例一样，只不过讨论的是多元回归的情形。

二元线性回归模型

$$y_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + e_i$$

$$e_i \sim N(0, \sigma_i^2)$$

$$\sigma_i^2 = e^{\alpha_0 + \alpha_1 \ln(x_{i1}) + \alpha_2 \ln(x_{i2})}$$

(i) $\alpha_1 = \alpha_2 = 0$

(ii) $\alpha_1 = 2, \alpha_2 = 0$

(iii) $\alpha_1 = 0, \alpha_2 = 2$

(iv) $\alpha_1 = \alpha_2 = 1$

***=====begin=====**

```
clear
set memory 40m
set matsize 2000
capture program drop het
program het
quietly {
drop _all
set obs `1'
gen x1=uniform()
gen x2=1+3*invnorm(uniform())
gen e=invnorm(uniform())
gen u2=exp(2*ln(x1))*e
gen u3=exp(2*ln(x2))*e
gen u4=exp(ln(x1)+ln(x2))*e
gen y1=1+3*x1+5*x2+e
gen y2=1+3*x1+5*x2+u2
gen y3=1+3*x1+5*x2+u3
gen y4=1+3*x1+5*x2+u4

forvalues i=1/4 {
reg y`i' x1 x2
imtest,white
scalar z`i'=r(p)
}
```

```

}
end

forvalues i=10(10)80 {
  simulate "het `i'" z1 z2 z3 z4,rep(1000)
  mkmat _s*, mat(A)
  mat B=(nullmat(B),A)
}
svmat B
drop _sim*
quietly forvalue i=1/32 {
  gen a`i'=(B`i'>0.05)
}
sum a*, sep(4)
*=====begin=====

```

15.5 估计方法：WLS 与 GLS

一般而言，我们无法确切地知道产生异方差的真正原因和异方差的具体形式。因此，在实证分析过程中具体采用哪一种方法来处理异方差也并不像我们前面的理论分析那么简单明了。

如果采用WLS方法，无论我们采用何种权重，只要权重与干扰项不相关，加权最小二乘估计都是一致的。然而，如果我们所使用的权重是错误的，就可能产生两个结果，一是加权最小二乘估计量不具有有效性，二是据此得到的系数的标准差是不正确的。对于FGLS，除非我们对异方差的形式有一个比较清楚地认识，而且可以用少数的几个参数进行描述，不要过于热衷于该方法。对异方差的错误设定往往会使我们的估计结果出现很大的偏差。在实证文献中，采用稳健的最小二乘估计，用White 或其修正公式估计标准差得到了更多的青睐。

最后，我们通过一个简单的模拟分析来说明上面的几点结论。模拟数据的生成过程为：

$$y_i = 10 + 5x_i + u_i$$

其中， $u_i \sim N(0; i)$ 。生成数据的命令为

```

=====begin=====
clear
capt prog drop het
prog het
drop _all
set obs 100
gen x = invnorm(uniform())
gen z = _n
gen e = sqrt(z)*invnorm(uniform())
gen y = 10 + 5*x + e
quietly reg y x
end
=====end=====

```

显然，模型存在异方差，为了检验OLS 估计的无偏性，我们输入如下命令：

```

het
reg y x

```

我们发现x和常数项的系数都非常接近于其真实值。如果把以上过程重复进行1000次，并记录历次回归的系数值

```
simulate _b _se, reps(1000):het
```

将得到x 系数的平均值，更接近其真实值，而相应的标准差也都非常小。这表明，在存在异方

差的情况下，OLS仍然是无偏且一致的。

接着，我们说明在采用WLS 回归时，权重的选择对回归结果的影响。首先，我们选择正确的权重：

```
=====begin=====
clear
capt prog drop het
prog het
drop _all
set obs 100
gen x = invnorm(uniform())
gen z = _n
gen e = sqrt(z)*invnorm(uniform())
gen y = 10 + 5*x + e
reg y x [aweight=1/z]
end
=====end=====
```

可以看出，系数的标准差明显减小了，而同时拟合优度 R^2 提高。将此过程重复进行1000次

```
simulate _b _se, reps(1000) :het
```

得到大样本下系数的标准差，明显低于我们采用OLS 估计时得到的大样本下的标准差。这说明在我们能够正确设定权重的情况下，WLS的确比OLS有效。然而，当我们错误设定权重时：

```
=====begin=====
clear
capt prog drop het
prog het
drop _all
set obs 100
gen x = invnorm(uniform())
gen z = _n
gen e = sqrt(z)*invnorm(uniform())
gen y = 10 + 5*x + e
reg y x [aweight=1/x ]
end
=====end=====
```

WLS 在小样本下的结果非常糟糕，比如同时，将上述过程重复进行1000次

```
simulate _b _se, reps(1000): het
```

我们发现在大样本下WLS 是一致的，但不具有有效性。需要说明的是，WLS 估计的一致性并不依赖于权重的选择，这与我们上面给出的结论是一致的。

15.6 广义最小二乘估计与 FGLS

二元线性回归模型

- (1) 对原方程用OLS进行估计，得到残差项的估计 \hat{u}_i ，
- (2) 计算 $\ln(\hat{u}_i^2)$
- (3) 用 $\ln(\hat{u}_i^2)$ 对所有独立的解释变量进行回归，然后得到拟合值 \hat{g}_i
- (4) 计算 $\hat{h}_i = \exp(\hat{g}_i)$
- (5) 用 $1/\hat{h}_i$ 作为权重, 做WLS回归

$$y_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + e_i$$

$$e_i \sim N(0, \sigma_i^2)$$

$$\sigma_i^2 = e^{\alpha_0 + \alpha_1 \ln(x_{i1}) + \alpha_2 \ln(x_{i2})}$$

$$(i) \alpha_1 = \alpha_2 = 0$$

$$(ii) \alpha_1 = 2, \alpha_2 = 0$$

$$(iii) \alpha_1 = 0, \alpha_2 = 2$$

$$(iv) \alpha_1 = \alpha_2 = 1$$

```

clear
set obs 100
gen x1=uniform()
gen x2=1+3*invnorm(uniform())
gen e=invnorm(uniform())
gen u2=exp(2*ln(x1))*e
gen u3=exp(2*ln(x2))*e
gen u4=exp(ln(x1)+ln(x2))*e
gen y1=1+3*x1+5*x2+e
gen y2=1+3*x1+5*x2+u2
gen y3=1+3*x1+5*x2+u3
gen y4=1+3*x1+5*x2+u4

```

*当误差结构已知时，用GLS

```

reg y1 x1 x2
reg y2 x1 x2 [aw=1/x1]
reg y3 x1 x2 [aw=1/x2]
gen x12=x1*x2
reg y4 x1 x2 [aw=1/x12]

```

*当误差结构未知时，可行的FGLS回归

```

forvalues i=1/4 {
  reg y`i' x1 x2
  predict uh`i',res
  gen luh`i'=ln(uh`i'*uh`i')
  reg luh`i' x1 x2
  predict g`i'
  gen `i'=exp(`i')
  reg y2 x1 x2 [aweight=1/`i']
}

```

16 随机过程模拟

16.1 时间数据函数

STATA 软件以 1960 年元旦为第 0 天，相应地 1960 年 1 月 2 日为第 1 天。

help dates

----- Numerical value & interpretation -----

Format	Meaning	Value=-1	Value=0	Value=1
%tc	clock	31dec1959	01jan1960	01jan1960
23:59:59.999	00:00:00.000	00:00:00.001		
%td	days	31dec1959	01jan1960	02jan1960
%tw	weeks	1959w52	1960w1	1960w2
%tm	months	1959m12	1960m1	1960m2
%tq	quarters	1959q4	1960q1	1960q2
%th	half years	1959h2	1960h1	1960h2
%tg	generic	-1	0	1

*计算自己的精确年龄

```
di (mdy(10,26,2007)-mdy(3,27,1979))/365.25
```

*其他时间函数

```
sysuse sp500, clear
gen d=day(date)
gen w=week(date)
gen m=month(date)
gen q= quarter(date)
gen hy= halfyear(date)
gen y=year(date)
gen ndate1=mdy(m,d,y)
gen weekd=dow(date) //周几
gen yeard=doy(date) // 一年中的第几天
```

*范围选择

```
tsset date
list if tin(08jan2001,23jan2001) //选择1月8号到23号的数据
list if tin(,08jan2001)
```

*生成滞后与超前项

```
use http://fmwww.bc.edu/ec-p/data/wooldridge/nyse, clear
save nyse, replace
tsset t //设定时间变量
g r_1=return[_n-1] //生成滞后变量
g r1=L.return //生成滞后变量，与前一命令等价
g r4=L4.return //生成四阶滞后变量，与前一命令等价
list r* in 1/10
g f2=F2.return //生成超前二阶变量
```

```

g d1=D.return //生成差分变量
g s1=S.return //生成季节性滞后变量
list return f* d* d s in 1/10
reg return L.return L2.return S.price D(1/3).price //在回归中使用差分\滞后
g price1=(price[_n-1]+price[_n]+price[_n+1])/3 //移动平均,步长为3
egen price2=ma(price),nomiss t(3) //另一种算法,与上一命令等价
egen price31=ma(price),t(31) //做月移动平均线
tsline price price1 price31
tsline price price31 in 200/400
smooth 31 price, gen(price31r) //非线性平滑
list price31* in 100/110
tsline price price31r in 200/400
gen rough=price-price31 //求平滑后的残差并做图
tsline price rough

```

16.2 模拟白噪声及检验白噪声

满足如下条件的序列称为严平稳序列: \forall 正整数 $m, \forall t_1, t_2, \dots, t_m \in T, \forall$ 正整数 τ , 有

$$F_{t_1, t_2, \dots, t_m}(x_1, x_2, \dots, x_m) = F_{t_1+\tau, t_2+\tau, \dots, t_m+\tau}(x_1, x_2, \dots, x_m)$$

严平稳要求的条件比较苛刻, 只有当序列所有的统计性质都不随着时间的推移而发生变化时, 该序列才能被认为是严平稳的。

满足如下条件的序列称为宽平稳序列

- 1) $EX_t^2 < \infty, \forall t \in T$
- 2) $EX_t = \mu, \mu$ 为常数, $\forall t \in T$
- 3) $\gamma(t, s) = \gamma(k, k+s-t), \forall t, s, k$ 且 $k+s-t \in T$

宽平稳是使用序列的特征统计量来定义的一种平稳性。它认为序列的统计性质主要由它的低阶矩决定, 所以只要保证序列低阶矩平稳 (二阶), 就能保证序列的主要性质近似稳定。通常情况下, 严平稳 (低阶矩存在) 能推出宽平稳成立, 而宽平稳序列不能反推严平稳成立。但不存在低阶矩的严平稳序列不满足宽平稳条件, 例如服从柯西分布的严平稳序列就不是宽平稳序列; 另外当序列服从多元正态分布时, 宽平稳可以推出严平稳

纯随机序列也称为白噪声序列, 它满足如下两条性质 (1) $EX_t = \mu, \forall t \in T$

$$(2) \gamma(t, s) = \begin{cases} \sigma^2, & t = s \\ 0, & t \neq s \end{cases}, \forall t, s \in T$$

白噪声序列的各序列值之间没有任何相关关系, 为“没有记忆”的序列, 白噪声序列具有方差齐性, 根据马尔可夫定理, 最小二乘估计值是 BLUE 的。如果白噪声序列

独立同分布, 则称为独立白噪声。如果服从正态分布, 则为正态白噪声。

Barlett 定理:如果一个时间序列是纯随机的, 得到一个观察期数为 n 的观察序列, 那么该序列的延迟非零期的样本自相关系数将近似服从均值为零, 方差为序列观察期数倒数的正态分布 $\hat{\rho}_k \sim N(0, \frac{1}{n})$, $\forall k \neq 0$

对白噪声的假设检验正是根据该分布理论。原假设: 延迟期数小于或等于 m 期的序列值之间相互独立 $H_0: \rho_1 = \rho_2 = \dots = \rho_m = 0, \forall m \geq 1$

备择假设: 延迟期数小于或等于 m 期的序列值之间有相关性

H_1 : 至少存在某个 $\rho_k \neq 0, \forall m \geq 1, k \leq m$

构造 Q 统计量 $Q = n \sum_{k=1}^m \hat{\rho}_k^2 \sim \chi^2(m)$

或 LB 统计量 $LB = n(n+2) \sum_{k=1}^m \left(\frac{\hat{\rho}_k^2}{n-k} \right) \sim \chi^2(m)$

当检验统计量大于分位点, 或该统计量的 P 值小于给定的水平时, 则可以以一定的置信水平拒绝原假设, 认为该序列为非白噪声序列。否则不能显著拒绝序列为纯随机序列的假定。

从网上下载的模拟ARMA过程的命令**sim_arma**³, 然后模拟正态白噪声

```
set seed 1234
```

```
drawnorm x, n(691) sds(3) clear
```

```
set seed 1234
```

```
sim_arma y, sigma(3) //注意到x和y两个序列是完全一样的
```

```
tsline y //趋势图
```

```
hist y, bin(50) norm //直方图
```

```
sum y, d //统计该随机过程的分布特征
```

```
ttest y==0 //检验零均值
```

```
sdtest y==1 //检验方差是否为1
```

```
pnorm y, grid //标准正态分布检验
```

```
wntestb y //白噪声检验
```

```
wntestq y //白噪声检验
```

```
corrgram y //Q检验
```

```
ac y //自相关图检验
```

```
pac y //偏相关图检验
```

利用美国股市的真实数据 `nyse.dta` 做类似分析, 比较两者的差异, 股市收益是随机过程吗? 是高斯白噪声吗?

```
use nyse, clear
```

```
tsset t
```

³ Search `sim_arma.net`, 查看说明, 选择一个文件点击, 在弹出的窗口中选择 `click here to install`

```

rename return y
tsline y           //趋势图
hist y, bin(50) norm //直方图
sum y, d //统计该随机过程的分布特征
ttest y==0 //检验零均值
sdtest y==1 //检验方差是否为1
pnorm y, grid //标准正态分布检验
swilk y //正态分布检验
sktest y //正态分布检验
wntestb y //白噪声检验
wntestq y //白噪声检验
corrgram y //Q检验
ac y //自相关图检验
pac y //偏相关图检验
reg y L.y

```

16.3 模拟自回归过程 AR 并检验稳定性

具有如下结构的模型称为 p 阶自回归模型，简记为 $AR(p)$

$$\begin{cases} x_t = \phi_0 + \phi_1 x_{t-1} + \phi_2 x_{t-2} + \cdots + \phi_p x_{t-p} + \varepsilon_t \\ \phi_p \neq 0 \\ E(\varepsilon_t) = 0, \text{Var}(\varepsilon_t) = \sigma_\varepsilon^2, E(\varepsilon_t \varepsilon_s) = 0, s \neq t \\ Ex_s \varepsilon_t = 0, \forall s < t \end{cases}$$

特别当 $\phi_0 = 0$ 时，称为中心化 $AR(p)$ 模型

```

clear
set seed 1234
sim_arma y, ar(0.8) n(100)
set seed 1234
g x=y in 1
quietly forvalues i=2/100 {
  replace x=0.8*x[_n-1]+invnormal(uniform()) in `i'
}

```

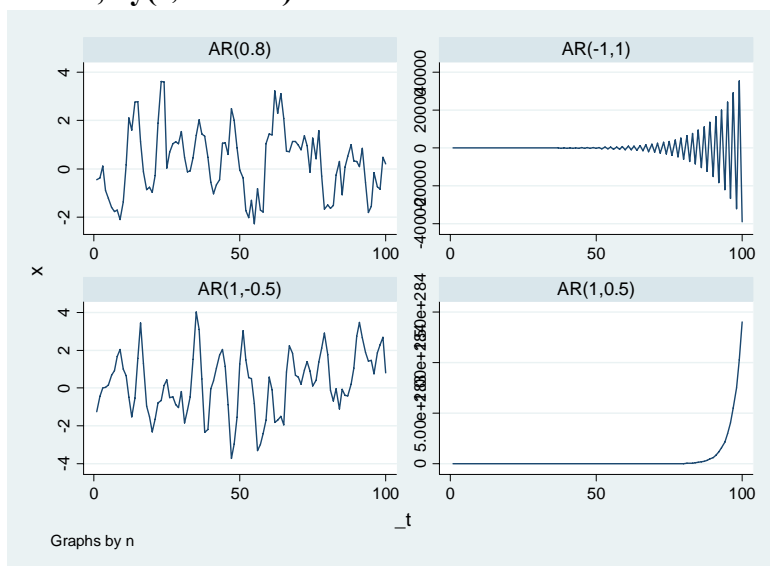
注意到上述的 y 和 x 的模拟结果完全一样。模拟如下的四个由正态白噪声构成的随机序列。

- (1) $x_t = 0.8x_{t-1} + \varepsilon_t$
- (2) $x_t = -1.1x_{t-1} + \varepsilon_t$
- (3) $x_t = x_{t-1} - 0.5x_{t-2} + \varepsilon_t$


```

(4)  $x_t = x_{t-1} + 0.5x_{t-2} + \varepsilon_t$ 
clear
sim_arma x1, nobs(100) ar(.8)
sim_arma x2, ar(-1,1)
sim_arma x3, ar(1, -0.5)
sim_arma x4, ar(1, 0.5) spin(2000)
reshape long x, i(_t) j(n)
label define nlb 1 "AR(0.8)" 2 "AR(-1,1)" 3 "AR(1,-0.5)" 4 "AR(1,0.5)"
label value n nlb
tsline x, by(n, rescale)

```



AR 模型并不都是平稳的，如何判别其平稳性呢？

特征根判别

AR(p)模型平稳的充要条件是它的p个特征根都在单位圆内，根据特征根和自回归系数多项式的根成倒数的性质，等价判别条件是该模型的自回归系数多项式的根都在单位圆外。

```

clear
sim_arma x1, nobs(100) ar(.8)
sim_arma x4, ar(1, 0.5) spin(2000)
kpss x1, notrend
kpss x4, notrend

```

平稳 AR 模型的统计性质

如果 AR(p)模型满足平稳性条件，则均值有 $Ex_t = E(\phi_0 + \phi_1 x_{t-1} + \dots + \phi_p x_{t-p} + \varepsilon_t)$

根据平稳序列均值为常数，且 $\{\varepsilon_t\}$ 为白噪声序列，推导出

$$Ex_t = \mu, E(\varepsilon_t) = 0, \forall t \in T \quad \mu = \frac{\phi_0}{1 - \phi_1 - \dots - \phi_p}$$

在平稳 AR(p)模型两边同乘 x_{t-k} ，再求期望

$$E(x_t x_{t-k}) = \phi_1 E(x_{t-1} x_{t-k}) + \cdots + \phi_p E(x_{t-p} x_{t-k}) + E(\varepsilon_t x_{t-k})$$

根据 $E(\varepsilon_t x_{t-k}) = 0$ 得协方差函数的递推公式 $\gamma_k = \phi_1 \gamma_{k-1} + \phi_2 \gamma_{k-2} + \cdots + \phi_p \gamma_{k-p}$

定义自相关系数 $\rho_k = \frac{\gamma_k}{\gamma_0}$ ，得自相关系数递推公式 $\rho_k = \phi_1 \rho_{k-1} + \phi_2 \rho_{k-2} + \cdots + \phi_p \rho_{k-p}$

AR 模型自相关系数具有拖尾性 $\rho(k) = \sum_{i=1}^p c_i \lambda_i^k$ c_1, c_2, \dots, c_p 不能恒等于零

和呈复指数衰减性 $\rho(k) = \sum_{i=1}^p c_i \lambda_i^k \rightarrow 0$

平稳 AR(1)模型的方差为

平稳 AR(1) 模型的协方差递推公式为 $\gamma_k = \phi_1 \gamma_{k-1} = \phi_1^k \gamma_0 = \phi_1^k \frac{\sigma_\varepsilon^2}{1 - \phi_1^2}$, $\forall k \geq 1$

AR(1)模型的自相关系数 $\rho_k = \phi_1^k, k \geq 0$

平稳 AR(2)模型的协方差函数递推公式和自相关系数分别为：

$$\left\{ \begin{array}{l} \gamma_0 = \frac{1 - \phi_2}{(1 + \phi_2)(1 - \phi_1 - \phi_2)(1 + \phi_1 - \phi_2)} \sigma_\varepsilon^2 \\ \gamma_1 = \frac{\phi_1 \gamma_0}{1 - \phi_2} \\ \gamma_k = \phi_1 \gamma_{k-1} + \phi_2 \gamma_{k-2}, \quad k \geq 2 \end{array} \right. \quad \rho_k = \left\{ \begin{array}{ll} 1, & k = 0 \\ \frac{\phi_1}{1 - \phi_2} & k = 1 \\ \phi_1 \rho_{k-1} + \phi_2 \rho_{k-2} & k \geq 2 \end{array} \right.$$

clear

sim_arma x1, nobs(100) ar(.8)

sim_arma x2, ar(-1.1)

sim_arma x3, ar(1, -0.5)

sim_arma x4, ar(1, 0.5) spin(2000)

ac x1,lags(20) title("ACF for AR(1) p= 0.8") ytitle("") ///

ylab(format(%3.1f)) note("") saving(1,replace)

ac x2,lags(20) note("") title("ACF for AR(1) p=-0.8") ///

ytitle("") ylab(format(%3.1f)) saving(2,replace)

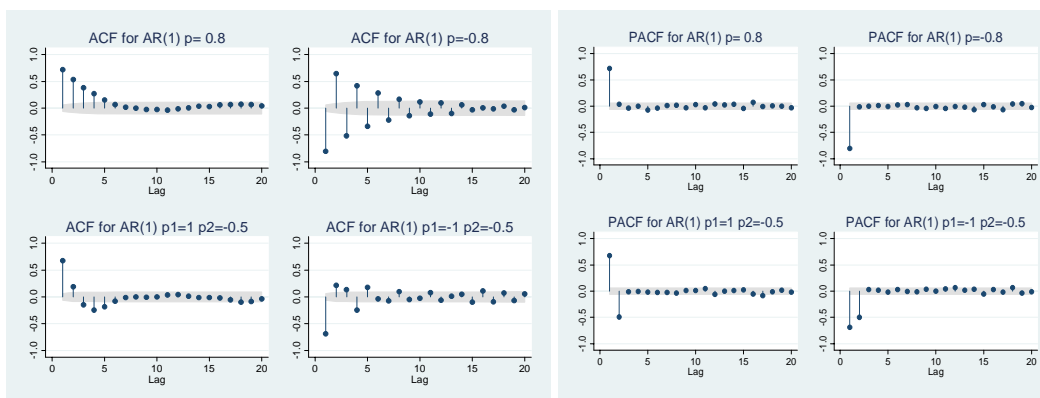
ac x3,lags(20) note("") title("ACF for AR(1) p1=1 p2=-0.5") ///

ytitle("") ylab(format(%3.1f)) saving(3,replace)

ac x4,lags(20) note("") title("ACF for AR(1) p1=-1 p2=-0.5") ///

ytitle("") ylab(format(%3.1f)) saving(4,replace)

graph combine 1.gph 2.gph 3.gph 4.gph,ycom



偏自相关系数

对于平稳 $AR(p)$ 序列, 所谓滞后 k 偏自相关系数就是指在给定中间 $k-1$ 个随机变量的条件下, 或者说, 在剔除了中间 $k-1$ 个随机变量的干扰之后, x_{t-k} 对 x_t 影响的相度量。用数学语言描述就是

$$\rho_{x_t, x_{t-k} | x_{t-1}, \dots, x_{t-k+1}} = \frac{E[(x_t - \hat{E}x_t)(x_{t-k} - \hat{E}x_{t-k})]}{E[(x_{t-k} - \hat{E}x_{t-k})^2]}$$

滞后 k 偏自相关系数实际上就等于 k 阶自回归模型第 k 个回归系数的值。

$$\begin{cases} \rho_1 = \phi_{k1}\rho_0 + \phi_{k2}\rho_1 + \dots + \phi_{kk}\rho_{k-1} \\ \rho_2 = \phi_{k1}\rho_1 + \phi_{k2}\rho_0 + \dots + \phi_{kk}\rho_{k-2} \\ \dots\dots\dots \\ \rho_k = \phi_{k1}\rho_{k-1} + \phi_{k2}\rho_{k-2} + \dots + \phi_{kk}\rho_0 \end{cases} \quad \phi_{kk} = \frac{E[(x_t - \hat{E}x_t)(x_{t-k} - \hat{E}x_{t-k})]}{E[(x_{t-k} - \hat{E}x_{t-k})^2]}$$

$AR(p)$ 模型偏自相关系数 P 阶截尾 $\phi_{kk} = 0, k > p$

16.4 模拟移动平均过程 MA

描述序列相关的最简单的线性过程是有限阶的白噪声移动平均得到的线性过程。

具有如下结构的模型称为 q 阶自回归模型, 简记为 $MA(q)$, 特别当 $\mu=0$ 时, 称为中心化 $MA(q)$ 模型

$$\begin{cases} x_t = \mu + \varepsilon_t - \theta_1 \varepsilon_{t-1} - \theta_2 \varepsilon_{t-2} - \dots - \theta_q \varepsilon_{t-q} \\ \theta_q \neq 0 \\ E(\varepsilon_t) = 0, \text{Var}(\varepsilon_t) = \sigma_\varepsilon^2, E(\varepsilon_t \varepsilon_s) = 0, s \neq t \end{cases}$$

MA 模型的统计性质

常数均值: $E x_t = E(\mu + \varepsilon_t - \theta_1 \varepsilon_{t-1} - \theta_2 \varepsilon_{t-2} - \dots - \theta_q \varepsilon_{t-q}) = \mu$

常数方差: $\text{Var}(x_t) = \text{Var}(\mu + \varepsilon_t - \theta_1 \varepsilon_{t-1} - \theta_2 \varepsilon_{t-2} - \dots - \theta_q \varepsilon_{t-q}) = (1 + \theta_1^2 + \dots + \theta_q^2) \sigma_\varepsilon^2$

自协方差与自相关系数 p 阶截尾

$$\gamma_k = \begin{cases} (1 + \theta_1^2 + \dots + \theta_q^2) \sigma_\varepsilon^2, & k = 0 \\ (-\theta_k + \sum_{i=1}^{q-k} \theta_i \theta_{k+i}) \sigma_\varepsilon^2, & 1 \leq k \leq q \\ 0, & k > q \end{cases}$$

MA(1)与 MA(2)的相关系数

$$\rho_k = \begin{cases} 1 & ,k=0 \\ \frac{-\theta_1}{1+\theta_1^2} & ,k=1 \\ 0 & ,k \geq 2 \end{cases} \quad \rho_k = \begin{cases} 1 & ,k=0 \\ \frac{-\theta_1 + \theta_1\theta_2}{1+\theta_1^2 + \theta_2^2} & ,k=1 \\ \frac{-\theta_2}{1+\theta_1^2 + \theta_2^2} & ,k=2 \\ 0 & ,k \geq 3 \end{cases}$$

偏自相关系数拖尾

$$\phi_{kk} = (-\theta_1\varepsilon_{t-1} - \dots - \theta_q\varepsilon_{t-q})(-\theta_1\varepsilon_{t-k-1} - \dots - \theta_q\varepsilon_{t-k-q+1})$$

$\theta_1, \dots, \theta_q$ 不恒为零 $\Rightarrow \phi_{kk}$ 不会在有限阶之后恒为零

例:

$$\begin{aligned} (1)x_t &= \varepsilon_t - 2\varepsilon_{t-1} \\ (2)x_t &= \varepsilon_t - 0.5\varepsilon_{t-1} \\ (3)x_t &= \varepsilon_t - \frac{4}{5}\varepsilon_{t-1} + \frac{16}{25}\varepsilon_{t-2} \\ (4)x_t &= \varepsilon_t - \frac{5}{4}\varepsilon_{t-1} + \frac{25}{16}\varepsilon_{t-2} \end{aligned}$$

MA 模型自相关系数的不唯一性, 上例中不同的 MA 模型可能具有完全相同的自相关系数和偏自相关系数。

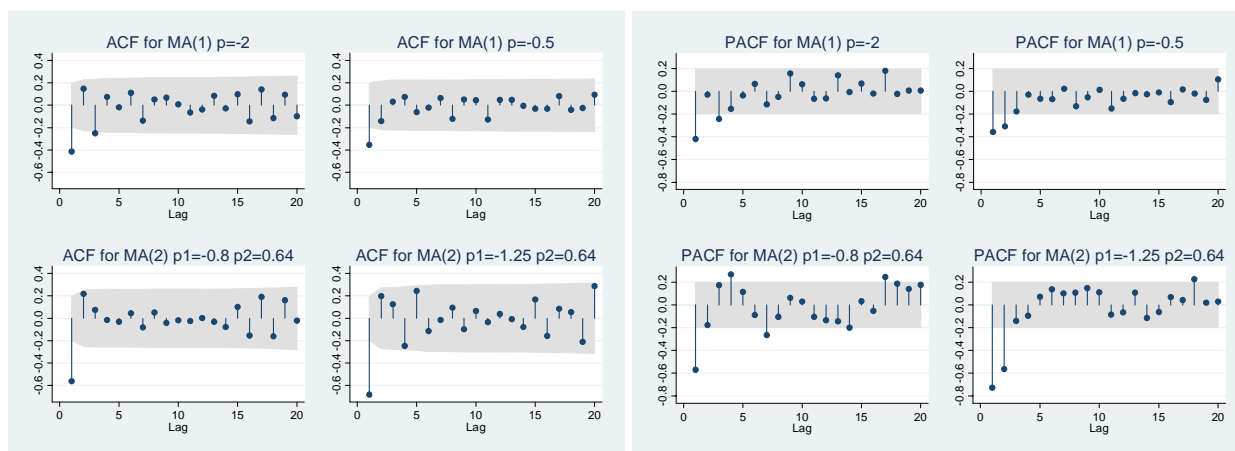
$$\begin{aligned} (1)x_t &= \varepsilon_t - 2\varepsilon_{t-1} \leftrightarrow (2)x_t = \varepsilon_t - 0.5\varepsilon_{t-1} \\ (3)x_t &= \varepsilon_t - \frac{4}{5}\varepsilon_{t-1} + \frac{16}{25}\varepsilon_{t-2} \leftrightarrow (4)x_t = \varepsilon_t - \frac{5}{4}\varepsilon_{t-1} + \frac{25}{16}\varepsilon_{t-2} \end{aligned}$$

可逆 MA 模型定义: 若一个 MA 模型能够表示称为收敛的 AR 模型形式, 那么该 MA 模型称为可逆 MA 模型。可逆概念的重要性: 一个自相关系数列唯一对应一个可逆 MA 模型。

MA(q)模型的可逆条件是: MA(q)模型的特征根都在单位圆内, $|\lambda_i| < 1$

等价条件是移动平滑系数多项式的根都在单位圆外 $\left| \frac{1}{\lambda_i} \right| > 1$

```
clear
sim_arma x1, nobs(100) ma(-2)
sim_arma x2, ma(-0.5)
sim_arma x3, ma(-0.8, 0.64)
sim_arma x4, ma(-1.25, 0.64)
pac x1, saving(1,replace)
pac x2, saving(2,replace)
pac x3, saving(3,replace)
pac x4, saving(4,replace)
graph combine 1.gph 2.gph 3.gph 4.gph,ycom
```



具有如下结构的模型称为自回归移动平均模型，简记为 $ARMA(p,q)$ ，特别当 $\phi_0 = 0$ 时，称为中心化 $ARMA(p,q)$ 模型。

$$\begin{cases} x_t = \phi_0 + \phi_1 x_{t-1} + \cdots + \phi_p x_{t-p} + \varepsilon_t - \theta_1 \varepsilon_{t-1} - \cdots - \theta_q \varepsilon_{t-q} \\ \phi_p \neq 0, \theta_q \neq 0 \\ E(\varepsilon_t) = 0, \text{Var}(\varepsilon_t) = \sigma_\varepsilon^2, E(\varepsilon_t \varepsilon_s) = 0, s \neq t \\ E x_s \varepsilon_t = 0, \forall s < t \end{cases}$$

$$\Phi(B) = 0$$

$ARMA(p,q)$ 模型的平稳条件: P 阶自回归系数多项式 $\Phi(B) = 0$ 的根都在单位圆外, 即 $ARMA(p,q)$ 模型的平稳性完全由其自回归部分的平稳性决定。

$ARMA(p,q)$ 模型的可逆条件: q 阶移动平均系数多项式 $\Theta(B) = 0$ 的根都在单位圆外, 即 $ARMA(p,q)$ 模型的可逆性完全由其移动平滑部分的可逆性决定。

直观地考察该模型自相关系数和偏自相关系数的性质。

模型	自相关系数	偏自相关系数
AR(P)	拖尾	P 阶截尾
MA(q)	q 阶截尾	拖尾
ARMA(p,q)	拖尾	拖尾

```
clear
sim_arma y1,ma(0.9) n(1000) time(t)
sim_arma y2,ar(0.1)
sim_arma y3,ma(0.2,0.3,0.8)
sim_arma y4,ma(0.2) ar(-0.7)
tsset t
ac y1,recast(bar) //自相关检验
pac y1,recast(bar) //偏自相关检验
arima y1,ma(1)
```

```

arima y1,arima(0,0,1) //模型拟合
ac y3,recast(bar) //自相关检验
pac y3,recast(bar) //偏自相关检验
arima y3,arima(0,0,3)
arima y3,ma(1/3)
predict uh3, resid //预测
corrgram uh3, lags(8) //残差检验
predict yh3
tsline y3 yh3 in 100/140
arima y2,arima(0,0,1)
arima y4,arima(1,0,1)

```

样本相关系数的近似分布

$$\text{Barlett} \quad \hat{\rho}_k \sim N(0, \frac{1}{n}), n \rightarrow \infty \quad \text{Quenouille} \quad \hat{\phi}_{kk} \sim N(0, \frac{1}{n}), n \rightarrow \infty$$

95%的置信区间

$$\Pr\left(-\frac{2}{\sqrt{n}} \leq \hat{\rho}_k \leq \frac{2}{\sqrt{n}}\right) \geq 0.95$$

$$\Pr\left(-\frac{2}{\sqrt{n}} \leq \hat{\phi}_{kk} \leq \frac{2}{\sqrt{n}}\right) \geq 0.95$$

模型定阶的经验方法:如果样本(偏)自相关系数在最初的 d 阶明显大于两倍标准差范围,而后几乎 95%的自相关系数都落在 2 倍标准差的范围以内,而且通常由非零自相关系数衰减为小值波动的过程非常突然。这时,通常视为(偏)自相关系数截尾。截尾阶数为 d 。

待估参数 $p+q+2$ 个未知参数 $\phi_1, \dots, \phi_p, \theta_1, \dots, \theta_q, \mu, \sigma_\varepsilon^2$

模型的显著性检验:目的是检验模型的有效性(对信息的提取是否充分),检验对象主要是残差序列。判定原则:一个好的拟合模型应该能够提取观察值序列中几乎所有的样本相关信息,即残差序列应该为白噪声序列,反之,如果残差序列为非白噪声序列,那就意味着残差序列中还残留着相关信息未被提取,这就说明拟合模型不够有效。

原假设:残差序列为白噪声序列

$$H_0: \rho_1 = \rho_2 = \dots = \rho_m = 0, \forall m \geq 1$$

$$H_1: \text{至少存在某个 } \rho_k \neq 0, \forall m \geq 1, k \leq m$$

备择假设:残差序列为非白噪声序列

LB 统计量 $LB = n(n+2) \sum_{k=1}^m \left(\frac{\hat{\rho}_k^2}{n-k} \right) \sim \chi^2(m)$

参数显著性检验:检验每一个未知参数是否显著非零 $T = \sqrt{n-m} \frac{\hat{\beta}_j - \beta_j}{\sqrt{a_{jj} Q(\hat{\beta})}} \sim t(n-m)$

同一个序列可以构造两个拟合模型,两个模型都显著有效,那么到底该选择哪个模型用于统计推断呢?一般采用最小信息量准则(An Information Criterion),该准则

力求使似然函数值越大越好，未知参数的个数越少越好。

AIC 统计量 $AIC = n \ln(\hat{\sigma}_\varepsilon^2) + 2(\text{未知参数个数})$

AIC 准则的缺陷：在样本容量趋于无穷大时，由 AIC 准则选择的模型不收敛于真实模型，它通常比真实模型所含的未知参数个数要多

SBC 统计量 $SBC = n \ln(\hat{\sigma}_\varepsilon^2) + \ln(n)(\text{未知参数})$

```
clear
set obs 200
set seed 1234
sim_arma y, ar(0.9) ma(0.9,0.3,0.5)
arima y, ar(1) ma(1/3)
icomp //该命令需要下载net get icomp
```

16.5 序列相关性检验

如由于消费习惯被包含在随机误差项中,出现序列相关性,省掉某些变量,如边际成本函数中省掉二次项,出现序列相关性。序列相关性导致参数估计量无效,变量显著性检验失去意义,预测失效。我们用真实的数据和模拟的数据进行对比检验。

```
use http://fmwww.bc.edu/ec-p/data/wooldridge/phillips, clear
tsset year, yearly
set seed 1234
sim_arma u, ar(0.8)
drawnorm x, mean(5) sds(9)
g y=20*x+u
reg y x
dwstat //dw检验一般是一阶序列相关
durбина //而durбина和bgodfrey可针对多阶序列相关
bgodfrey, lag(1/3)
predict e, res
g el=e[_n-1]
line e el, sort
reg e x L(1/3).e
dwstat
bgodfrey, lag(1/3)
prais y x
prais y x, corc
prais y x, twostep
prais y x, sssearch
prais y x, robust

reg inf unem
predict uh, res
```

```

reg uh L.uh
reg inf unem
dwstat
durbina
prais inf unem, ssearch

```

wooldbridge 书上的例子(12.3), 配合模拟数据理解

```

use http://fmwww.bc.edu/ec-p/data/wooldridge/barium, clear
tsset t, yearly
reg lchnimp lchempi lgas lrtwex befile6 affile6 afdec6
predict uh, res
reg uh lchempi lgas lrtwex befile6 affile6 afdec6 L(1/3).uh
test L1.uh L2.uh L3.uh
prais lchnimp lchempi lgas lrtwex befile6 affile6 afdec6, corc
use http://fmwww.bc.edu/ec-p/data/wooldridge/PRMINWGE, clear //另一个实例
newey lprepop lmincov lprgnp lusgnp t, lag(2)
reg lprepop lmincov lprgnp lusgnp t
bgodfrey
bgodfrey, lags(1 2 3)
dwstat
drawnorm x, mean(5) sds(9) clear
sim_arma u, ar(-0.2 -0.8) //模拟多阶序列相关性数据
g y=20*x+u
reg y x
dwstat
durbina, lag(1 2 3) robust
bgodfrey, lag(1/5)
predict e, res
reg e L(1/2).e
reg e L(1/5).e
newey y x, lag(2)
prais y x, ssearch

```

16.6 单位根检验

模拟一个不含漂移的随机游走,并考察其特征

```

clear
sim_arma y, ar(1) n(691) spin(2000) time(t)
tsline y D.y //随机游走: 一阶水平序列与一阶差分序列
pac y
ac y
reg y L.y //单位根检验方法1, 直接回归, t检验. 即由 $Y_t = \rho Y_{t-1} + u$ , 检验 $\rho = 1$ 
ttest y = L //即由 $Y_t = \rho Y_{t-1} + u$ , 检验 $\rho = 1$ 
reg D.y L.y /*单位根检验方法2, 变形 $\Delta Y = (\rho - 1) Y_{t-1} + u$ , 即检验 $\rho = \rho - 1 = 0$ ?

```


由于在单位根下, p *统计量不服从 t 分布,而是服从一个特殊的分布(Dickey-Fuller distribution),因此以上检验不正确,正确的方法为: */

dfuller *y*, **lags**(0) //DF检验

dfuller *y*, **lags**(0) **regress** //DF检验同时报告 $\Delta Y = (p-1) * Y_{t-1} + u$ 的回归结果

dfgls *y* //另外两种检验方法

dfgls *y*, **maxlag**(3)

pperron *y*

pperron *y*, **reg**

利用真实数据NYSE进行分析,比较模拟数据和真实数据,股价是随机游走吗?

use *nyse*, **clear**

tsset *t*

*随机游走:一水平序列与一阶差分序列

corr *price* *L2.price*

tsline *price* *D.price*

corrgram *price*, **lag**(10)

pac *price*

ac *price*

dfuller *price*

dfuller *price*, **lags**(0) **regress**

dfgls *price*

pperron *price*

xcorr *price* *return*, **lags**(9)

xcorr *price* *return*, **lags**(9) **table**

sim_arma *y*, **ar**(1) **n**(691) **spin**(2000) **time**(*t*)

replace *y*=*y*+50

tsline *price* *D.price* || **line** *y* *D.y*

corrgram *y*

corrgram *price*

dfuller *y*, **lags**(0) **regress**

dfuller *price*, **lags**(0) **regress**

/*注意一个区别: 设 y 的均值为 u ,当 u 已知时,将序列平移 u 即得到零均值序列;若 u 未知,则由 $y-u=p(y_{-1}-u)+e$ 整理得到 $y=A+py_{-1}+e$ 其中 $A=u(1-p)$,若存在单位根,即 $p=1$,则 $A=0$ */

reg *y* *L.y*

*模拟一个含有漂移项的随机游走,并进行单位根检验

use *nyse*, **clear**

tsset *t*

g *y*=49.75 **in** 1 //使模拟的期初值与真实值一样

quietly forvalues *i*=2/691 {

replace *y*=0.2+*y*[*n*-1]+**invnorm**(**uniform**()) **in** '*i*'

}

tsline *y* *D.y* || **tsline** *price* *D.price*

*对比检验: 若 $y=b+y_{t-1}+e$, 则 $y-a-b*t=p(y_{t-1}-a-b*t)+e$,整理得 $y=A+Bt+py_{t-1}+e$,其中

$A=a(1-p)+bp, B=b(1-p)$, 若 $p=1$, 则 $A=b, B=0$

```
reg y L.y
reg D.y L.y
reg D.y L.y t
dfuller y, lags(0) regress
dfuller price, lags(0) regress
dfuller y, lags(0) trend regress
dfuller price, lags(0) trend regress
```

/*协整检验:设 x 和 y 有长期均衡关系 $y_t=a+bx_t+u_t$,则 u_t 叫非均衡误差,应为一平稳时间序列,具有零期望值,即为 $I(0)$;假设 X 与 Y 是非稳定的时间序列,但其线性组合是平稳的 $u_t=y_t-a-bx_t$,则称 X 与 Y 是协整的.*/

```
clear
sim_arma x, ar(1) n(200) time(t) spin(2000)
g y=10+0.5*x+invnormal(uniform())
tsline x y
reg y x
predict e, res
dfuller y
dfuller x
dfuller e, regress
```

伪回归问题模拟

```
clear
capt prog drop bb
prog bb
drop _all
set obs 100
sim_arma x, ar(0.8)
sim_arma y, ar(0.8)
reg y x
test x=0
scalar p1=r(p)
prais y x, corc
test x=0
scalar p2=r(p)
end

quietly simulate p1=p1 p2=p2, reps(1000):bb
gen h1=p1<0.05
g h2=p2<0.05
tab h1
tab h2
*hist h, xline(0.05)
```

16.7 平滑分析

```

use nyse, clear
tsset t
rename price y
tssmooth ma y1=y, window(4 0 3) replace
/*移动平均, 其中window中的第一个数字表示滞后几步, 中间为是否包括原观察
值, 后面为向前移动几步*/
tssmooth ma y2=y, weight(5 1 7 <2> 8) replace
/*移动平均, weight中<>之前的数字表示对滞后加权的权数, <>中为对当期值的
权重, <>后数据为向前移动权重*/
tssmooth exponential y1=y, parms(0.1) replace //指数平滑
tssmooth exponential y2=y, parms(0.9) replace
tsline y y1 y2 in 500/600

tssmooth dexponential y1=y, parms(0.1) replace
tssmooth dexponential y2=y, parms(0.9) replace
tsline y y1 y2 in 500/600

tssmooth dexponential y1=y in 500/680, forecast(10) replace //预测
tssmooth exponential y2=y in 500/680, parms(0.5) forecast(10) replace
tsline y y1 y2 in 650/l

tssmooth hwinters y1=y in 500/680, p (0.3 0.2) f (10) replace //Holt-Winters平滑
tssmooth h y2=y in 500/680, p (0.1 0.9) f (10) replace
tsline y y1 y2 in 650/l

*****Holt-Winters season smoothing
tssmooth shwinters y1=y in 500/680, p (0.3 0.2 .1) period(4) f (10) replace
tssmooth s y2=y in 500/680, p (0.1 0.9 .2) f (10) per (4) replace //HW季节平滑
tsline y y1 y2 in 650/l

tssmooth nl y1=y in 500/680, smother(3rssh) replace
tssmooth nl y2=y in 500/680, smother(4253h, twice) replace
tsline y y1 y2 in 650/l

```

17 计量经济学基本理论模拟

17.1 经典假设满足时 OLS 估计量的小样本性质

当经典假设 1-6 均满足时，OLS 估计量无偏、具有最小方差且服从精确的正态分布。

```

=====begin=====
clear
set memory 40m
mat m=(3,4,5,0)
mat sd=(9,9.6,0.01,0\9.6,16,12,0\0.01,12,25,0\0,0,0,1)
/*生成假设的数据集,该数据集中能力X1与教育X2高度相关(不是完全相关),能力X1与经
验X3不相关,教育X2与经验X3相关,所有变量与相貌x4都不相关,无完全共线假设成立*/
drawnorm x1 x2 x3 x4,n(10) means(m) cov(sd)
gen u=invnorm(uniform()) /*设随机误差u服从正态分布 $u \sim N(0, 1)$ ,则误差服
从正态分布假设满足。与前一个命令语句一起从
同一个总体(表现为同一种数据生成机制中进行
简单随机抽样,样本为10个,样本独立同分布,
故满足随机抽样假设(IID)*/
gen y=12+5*x1+10*x2+3*x3+x4+u /*假设的真实情形,其中10为已知真实参数,在后面的
分析中,我们要通过样本来估计这些总体参数,
即教育的回报10。又因为我们在上述生成u的过程
中,没有考虑任何X的信息。完全随机地产生了u,
因此零条件误差假设和条件同方差假设成立,即
 $E(u|X)=0$ ,  $Var(u|X)=9$ 。*/
reg y x1 x2 x3 x4 /*函数形式正确设定,并且满足参数线性关系,假
设成立,X2的真实值10是否在95%置信区间内?*/

/*在一次回归结果中,OLS估计结果与真实参数值之间可能存在较大差异,但是反复抽取
同样大小的样本,估计出若干个OLS估计值,这些估计值将服从正态分布。尽管相对于我们要
估计的6个未知参数而言,我们的样本量非常小(仅为20个),但是估计量具有非常优良的统计
性质*/
capt prog drop _all
prog bb
drop _all
drawnorm x1 x2 x3 x4,n(10) means(m) cov(sd)
gen u=invnorm(uniform())
gen y=12+5*x1+10*x2+3*x3+x4+u
quietly reg y x1 x2 x3 x4
end

simulate _b, reps(200): bb
sum _b_x2 //能非常明显地看出,OLS估计量是无偏的,均值近靠10
pnorm _b_x2 //正态分布的图形检验
swilk _b_x2 //正态分布检验,参数估计值服从正态分布
hist _b_x2
=====end=====

```

17.2 条件误差服从正态分布的假设不成立时 OLS 的小样本性质

当假设 1-5 均满足（即高斯-马尔科夫假设成立），但第六个假设（即条件误差服从正态分布）不满足时，且样本过小时，OLS 估计量将不再服从正态分布。

```

=====begin=====
drop _all
drawnorm x1 x2 x3 x4,n(10) means(m) cov(sd)
rndexp 20 1 //误差xe服从指数，则误差服从正态分布假设不满足
gen y=12+5*x1+10*x2+3*x3+x4+xe /*零条件均值假设和条件同方差假设成立，即
                                E(u|X)=0, Var(u|X)=9。*/
reg y x1 x2 x3 x4                                /*模型正确设定假设成立，X2的真实值10是否在
                                                    95%置信区间内？*/

```

/*在一次回归结果中，OLS估计结果与真实参数值之间可能存在较大差异，但是反复抽取同样大小的样本，估计出若干个OLS估计值，这些估计值满足无偏性，但并不服从正态分布。

```

*/
capt prog drop _all
prog bb
drop _all
drawnorm x1 x2 x3 x4,n(10) means(m) cov(sd)
rndexp 20 1 //误差服从正态分布假设不满足
gen y=12+5*x1+10*x2+3*x3+x4+xe
quietly reg y x1 x2 x3 x4
end
simulate _b, reps(200) :bb
sum //能非常明显地看出，OLS估计量仍是无偏的
hist _b_x2
pnorm _b_x2 //正态分布的图形检验，不服从正态分布
swilk _b* //正态分布检验，参数估计值不服从正态分布
=====end=====

```

17.3 条件误差服从正态分布假设不成立时 OLS 的大样本性质

当假设 1-5 均满足，但第 6 个假设（条件误差服从正态分布）不满足，然而样本很大时（如大于 100），OLS 估计量服从渐近正态分布。而且估计量是**一致的**。

```

=====begin=====
drop _all
drawnorm x1 x2 x3 x4,n(10000) means(m) cov(sd)
rndexp 10000 1 //误差xe服从对数正态分布，则误差服从正态分布假设不满足
gen y=12+5*x1+10*x2+3*x3+x4+xe /*模型正确设定假设，零条件均值假设和条件同方差假设成立*/
reg y x1 x2 x3 x4                                /*X2的真实值10是否在95%置信区间内？*/
/*在一次回归结果中，OLS估计结果与真实参数值之间可能存在较大差异，但是反复抽取同样大小的样本，估计出若干个OLS估计值，这些估计值近似服从正态分布，随着样本容量增大，越接近正态分布。*/
capt prog drop _all
prog bb

```

```

drop _all
drawnorm x1 x2 x3 x4,n(10000) means(m) cov(sd) //大样本
rndexp 10000 1 //误差xe服从对数正态分布, 则误差服从正态分布假设不满足
gen y=12+5*x1+10*x2+3*x3+x4+xe
quietly reg y x1 x2 x3 x4
end
simulate _b, reps(200) :bb
sum _b_x2 //能非常明显地看出, OLS估计量是无偏的
qnorm _b_x2 //正态检验图, 近似服从正态分布
swilk _b* //正态分布检验, 参数估计值服从正态分布

```

*******end*******

综合: 重复执行下面的程序, 通过对比第六个假设(误差服从正态分布)成立和不成立时, 估计量的无偏性和正态性, 我们可进一步发现, 当假设1-5均满足, 但第6个假设不满足时, 且样本过小时, OLS估计量将不再服从正态分布。

*******begin*******

```

mat m=(3,4,5,0)
mat sd=(9,9.6,0.01,0\9.6,16,12,0\0.01,12,25,0\0,0,0,1)

capt prog drop _all
prog bb
drop _all
drawnorm x1 x2 x3 x4,n(20) means(m) cov(sd) //小样本
gen u=invnorm(uniform()) //误差项正态分布
gen y=12+5*x1+10*x2+3*x3+x4+u
quietly reg y x1 x2 x3 x4
end

capt prog drop bb2
prog bb2
drop _all
drawnorm x1 x2 x3 x4,n(20) means(m) cov(sd) //小样本
rndexp 20 1 //误差项非正态分布
gen y=12+5*x1+10*x2+3*x3+x4+xe
quietly reg y x1 x2 x3 x4
end

capt prog drop bb3
prog bb3
drop _all
drawnorm x1 x2 x3 x4,n(10000) means(m) cov(sd) //大样本条件
rndexp 10000 1 //误差项非正态分布
gen y=12+5*x1+10*x2+3*x3+x4+xe
quietly reg y x1 x2 x3 x4
end

simulate _b, reps(200) :bb
mkmat _b_x2, mat(A)

simulate _b, reps(200): bb2
mkmat _b_x2, mat(B)
mat A=(nullmat(A),B)

```

```

simulate _b, reps(200) : bb3
mkmat _b_x2, mat(B)
mat A=(nullmat(A),B)
svmat A

pnorm A1 //当误差项服从正态分布时，小样本的OLS估计值服从正态分布
pnorm A2 //当误差项不服从正态分布时，小样本的OLS估计值不服从正态分布
pnorm A3 //当误差项不服从正态分布时，小样本的OLS估计值近似服从正态分布
swilk A* //正态性的统计检验
sum A* //所有的估计都是无偏的
=====end=====

```

17.4 第一假设不成立时

当模型正确设定假设不成立时，即使其他假设均成立，且样本很大，OLS 估计量也是有偏的，不一致的。

```

=====begin=====
*函数形式误设
capt prog drop _all
prog bb
drop _all
drawnorm x1 x2 x3 x4, n(100) means(m) cov(sd)
gen u=invnorm(uniform())
gen y=12+5*x1+10*x2+5*x2^2+3*x3+x4+u
quietly reg y x1 x2 x3 x4 //误将平方项遗漏
end
simulate _b, reps(200) : bb
sum _b_z //当函数形式误设时，OLS估计量是有偏的，不一致的
=====end=====

=====begin=====
*函数形式误设
drawnorm x1 x2 x3 x4, n(100) means(m) cov(sd)
gen u=invnorm(uniform())
gen y=12+5*x1+10*x2+5*x2^2+3*x3+x4+u
quietly reg y x1 x2 x3 x4 //误将平方项遗漏
ovtest //设定检验
ovtest, rhs
gen z=x2^2
quietly reg y x1 x2 z x3 x4 //误将平方项遗漏

sum _b_z //当函数形式误设时，OLS估计量是有偏的，不一致的
=====end=====

```

17.5 第二假设不成立时

当随机抽样假设不成立时，即使其他假设均成立，且样本很大，OLS 估计量也是有偏的，不一致的。

```

=====样本选择=====begin=====
drop _all
drawnorm x2 x3 x4,n(1000) means(m) cov(sd)
gen u=3*invnorm(uniform())
gen y=5*x1+10*x2+3*x3+x4+u /*正确的函数形式*/
sort y
keep in 1/50 /*样本是非随机抽取的，随机抽样假设不满足*/
reg y x1 x2 x3 x4 /*真实值10是否在95%置信区间内*/
=====end=====

=====truncated model=====begin=====
drop _all
drawnorm x2,n(1000)
gen u=9*invnorm(uniform())
gen y=-5+10*x2+u /*正确的函数形式*/
drop if y<0
reg y x2 /*真实值10是否在95%置信区间内*/
truncreg y x2, ll(0)
=====end=====

=====sensored model=====begin=====
drop _all
drawnorm x2,n(1000)
gen u=9*invnorm(uniform())
gen y=-5+10*x2+u /*正确的函数形式*/
replace y=0 if y<0
reg y x2 /*真实值10是否在95%置信区间内*/
truncreg y x2, ll(0)
=====end=====

=====异常值=====begin=====
drop _all
drawnorm x2,n(1000)
gen u=9*invnorm(uniform())
gen y=-5+10*x2+u /*正确的函数形式*/
replace y=-10000 in 100
reg y x2 /*真实值10是否在95%置信区间内*/
rreg y x2
=====end=====

```


17.6 第三假设不成立时

当零条件误差假设不成立时，即使其他假设均成立，且样本很大，OLS 估计量也是有偏的，不一致的。

```

*=====begin=====
*遗漏变量：被遗漏的变量与其他解释变量相关 因此使得 $E(u|x) \neq 0$ 
capt prog drop_all
prog bb
drop_all
drawnorm x1 x2 x3 x4,n(100) means(m) cov(sd)
gen u=3*invnorm(uniform())
gen y=12+5*x1+10*x2+3*x3+x4+u
quietly reg y x2 x3 x4 //能力x1被遗漏
end
simulate _b, reps(200) :bb
sum //当存在遗漏变量时，OLS估计量是有偏的

*遗漏变量：被遗漏的变量与所有的其他解释变量均不相关
capt prog drop_all
prog bb
drop_all
drawnorm x1 x2 x3 x4,n(100) means(m) cov(sd)
gen u=invnorm(uniform())
gen y=12+5*x1+10*x2+3*x3+x4+u
quietly reg y x1 x2 x3 //相貌x4被遗漏
end
simulate _b, reps(200) : bb
sum //当被遗漏变量与其他自变量均不相关时，OLS估计量是无偏的
*=====end=====

```

17.7 第四假设不成立时

当无完全共线性假设不成立时，即存在完全的共线性时，某些变量的系数根本无法估计。

```

*=====begin=====
drop_all
drawnorm x1 x2 x3 x4,n(100) means(m) cov(sd)
gen u=3*invnorm(uniform())
gen y=12+5*x1+10*x2+3*x3+x4+u
gen x5=3*x2+x1
reg y x1 x2 x3 x4 x5 //STATA将自动去掉一个变量
*=====end=====

```

17.8 第五假设不成立时（略）

当条件同方差假设不成立时，即存在序列相关或异方差，相关模拟见第 15 和第 16 讲。

18 计量经济学综合案例⁴

18.1 简单回归分析

/*问题：班级规模大小对学生成绩有什么影响？

为什么要讨论该问题：教育部门和家长希望多雇佣老师，希望班级不要太拥挤，因为他们认为“班级越小，越有利于提高学生的成绩。”但是纳税人不乐意为此多掏钱，没有孩子的公众也不希望更多的公共资源被用于增加教师，减少班级规模。他们认为：“班级规模大小与学生的成绩无关”。于是争论的关键证据（事实）是：班级规模是否影响学生成绩？

问题的操作化：如果将班级规模平均减少 1 名学生，那么学生在标准化考试中成绩会变化多少？

模型设定： $\text{testscore} = a + b * \text{classsize} + u$

数据收集：caschool.dta

该数据为加州所有 1998-1999 年 420 个 K-6 和 K-8 地区，考试成绩是斯坦福 9 类达标考试中关于阅读和数学成绩的平均数。学校特征（地区平均）包括注册人数、老师数量、每间教室的计算机数量和每个学生的费用支出。其中生师比为该地区全职老师等量人数除以学生人数。学生的人口统计量包括：在公共援助计划中受援助学生百分比，享有减价午餐学生百分比，以及学习英语的学生非分比。*/

```
*****begin*****

*计算过程：简单回归分析
clear
cap log close
*****
* stock 于 2006 年 12 月 24 日用于计算班级规模和考试成绩的简单回归模型程序
*****

log using scoresize.log,replace
set more off
*****

*修改路径以便在你自己的计算机上运行
cd F:\教学\STATA 讲义
use caschool.dta
*****数据中变量的含义*****

* dist_code -- 地区代码
* Read_scr  -- 平均阅读成绩
* Math_scr  -- 平均数学成绩
* County    -- 县
* District  -- 学区
* gr_span   -- 年级
* enr_tot   -- 总学生数
* teachers  -- 总老师数
* computer  -- 学校的计算机总数
* testscr   -- 平均考试成绩 (= (read_scr+math_scr)/2 )
* comp_stu  -- 生均计算机数 (= computer/enr_tot)
* expn_stu  -- 生均经费
```

⁴ 该案例来自于 stock 等《introduction to Econometrics》。

```

* str –生师比 (teachers/enrl_tot)
* el_pct – 英语学习者百分比
* Meal_pct – 享有午餐补助学生百分比
* clw_pct – 享有收入援助计划学生百分比
* aving – 地区平均收入(in $1000's)

***** 描述性统计：1998 年加州 420 个地区五年级学生考试成绩与师生比分布表 *****
sum str          //均值与标准差
pctile pct_tsc = testscr, nq(20) genp(pctx)    //计算 20 等分百分位数
pctile pct_str = str, nq(20)
list pctx pct_str pct_tsc in 1/20
**百分位数的含义：生师比的第 10 个百分位数是 17.3,即只有 10%的地区生师比低于 17.3.

*****相关分析*****
cor str testscr
*相关系数为-0.23,两个变量之间具有弱的负相关关系。

***** 绘图分析 *****
tw (scatter testscr str,sort) (lfit testscr str)

*****回归分析*****
reg testscr str, r
/*斜率为-2.28,意味着班级规模减少 1 名, 预测考试成绩会提高 2.28 分
这一结果是大还是小呢？假设有一个地区处在中位数水平, 即生师比为 19.7,考试成绩为
654.5 分, 该地区的教育主管将每个班减少 2 名学生, 这一变化将使生师比从第 50 个百分位移
动到第 10 个百分位数的位置, 成绩将增加 4.6 分, 也即使该地区的成绩从中位数上升到第 60
个百分位数。
当自变量为二元时的回归*/
gen d = (str<20)
reg testscr d, r

```

/*结论：根据 1998 年加州考试成绩数据集中的 420 个观测值, 我们的回归分析表明, 生师比与考试成绩之间存在负向的关系, 即班级规模较小的地区会有较高的考试成绩。从实际意义上说, 回归系数是比较大的：平均来看, 生师比每减少 2 个单位, 其考试成绩平均会提高 4.6 分, 这相应地把该地区从考试成绩分布的第 50 个非分位数位置, 上移到大约第 60 个百分位数的位置。

生师比这一回归系数统计上在 5% 的显著性水平下显著地异于 0.即潜在的随机样本中通过纯粹随机抽样的方法估计出该系数为 0 的概率极其小, 约为 0.001%。

可能的争议：虽然平均来看, 生师比较低的地区有较高的考试成绩, 但是这真的意味着降低生师比确实能提高学生成绩吗？或者说估计出的关系就是教育主管进行决策所需要的那个因果关系吗？

人们有理由担心降低生师比可能不是提高学生成绩的原因, 比如说加州有一些多民社区, 这些移民社区比较穷, 因此他们的班级规模比较大, 但是影响学生成绩更重要的原因是这些孩子的母语不是英语, 在英语标准化考试中他们的成绩自然难以与英语为母语的学生相比。

这些其他因素的影响意味着, 这种简单的 OLS 分析实际上对于决策几乎没有什么价值, 事实上它可能会造成误导：仅改变生师比, 而不改变那些真正决策孩子成绩的其他因素, 结果花了钱却不能产生任何好的效果。

18.2 多元回归分析

由于在简单回归分析中，只研究了生师比这一变量，忽略了一些潜在的可能决定考试成绩的重要因素，比如学校特征、老师素质和硬件。学生特征，家庭背景。根据经验，这些因素当然会影响到学生的考试成绩。在简单回归分析的总结中，曾提及一个变量，即移民人口，他们的孩子仍在学习英语，因此不考虑英语学习者的百分比这一变量会使我们的估计量有偏差。

教育主管正在考虑增加其所在学区的教师量，但却无法控制所在地区的移民人数的百分比，因此，教育主管仅对保持其他因素不变时的生师比对学生成绩的感兴趣（只改变我们所能改变的事）。换言之：我们集中关注具有可比性的地区（在该例子中是英语学习者非分比可比），而不是所有地区的数据。

计算过程：多元回归分析*/

```
use caschool.dta, clear
```

```
*****描述性分析与参数检验*****
```

/*先将生师比以20为界分为两组，低生师比组和高生师比组，再按英语学习者百分比分为四组，计算所有组的均值，并在零假设（高低生师比的学习成绩一样）下进行t检验。*/

```
gen str_20 = (str<20) //将生师比分为高低两组
```

```
gen ts_lostr = testscr if str_20==1
```

```
gen ts_histr = testscr if str_20==0
```

```
xtile elq=el_pct,nq(4)
```

```
table str_20 elq,c(mean testscr) format(%5.2f)
```

```
ttest ts_lostr=ts_histr, unpaired //高低生师比组学习成绩相等零假设的t检验
```

```
by elq,sort: ttest ts_lostr=ts_histr, unpaired //英语学习比例与大小班交互后成绩: t检验
```

```
anov testscr str_20 //方差分析
```

/*结果分析：在没有区分英语学习者时，低生师比地区比高生师比地区的平均考试成绩高7.4分，两者平均成绩是相同的零假设在1%的显著性水平下被拒绝。

按英语学习者百分比的四分位数分组后，证据显示了不同的结论。在英语学习者最少的地区，低生师比和高生师比只差1.3分，第二个四分位数组中，高4.3分，第三个四分位数组中，高出4.9分，而英语学习者最多的地区，平均成绩只差1.9分。也就是说，一旦保持英语学习者这个指标不变，生师比对学习成绩的影响只有总体估计值的一半或更少。

由于英语学习者少的地区倾向于有低的生师比和高的学习成绩，而高的英语学习者地区倾向于有高的生师比和低的学习成绩，因此简单的回归分析存在遗漏变量偏差问题，需要多元回归分析*/

```
*****多元回归分析*****
```

```
reg testscr str, r //简单回归分析可能存在遗漏变量偏差
```

```
esti store model1
```

```
reg testscr str el_pct, r //控制英语学习者百分比后的多元回归分析
```

```
esti store model2
```

/*结论：生师比的OLS估计值为-1.10,与简单回归的估计值-2.28相比，影响的程度变小了，回归分析比分组分析更优越：它提供了生师比减少1个单位对成绩影响的定量估计值，这是决策时需要的，它也易于扩展到多于两个控制变量的情形。*/

/*根据目前的证据，减小班级规模对提高成绩会有帮助，但是教育主管面临另一个更细微的问题：如果雇更多的教师，将不得不通过减少其他的预算开支（比如减少维护办公费用，减少电脑的购买），因为纳税人不愿意多花钱来请更多的老师，所以它必须在目前的费用限制下进行结构调整，因此教育主管想知道：在保持生均费用不变的情况下，减少生师比是否会对成绩

有积极影响? */

```
replace expn_stu = expn_stu/1000    //变换支出的单位
```

```
reg testscr str expn_stu el_pct, r
```

```
esti store model3
```

/*结论：保持生均费用和英语学习者百分比不变的情况下，改变生师比的影响非常小，仅-0.29,而且该系数的真实值为零的假设不能被拒绝。

于是教育主管决定不做结构性调整，而是要求追加教育预算，但是纳税人并不同意，他们断言生师比和生均费用对提高学习成绩都没有影响。这意味着在学习成绩的决定方程中，生师比和生均费用前的系数都为零。真的是这样吗? */

```
test str expn_stu
```

/*结论：F统计量为5.43,零假设在1%的显著性水平下被拒绝。所以我们可以拒绝纳税人提出的“生师比和生均费用对学习成绩都没有影响的”假设。*/

```
*****相关分析 *****
```

```
cor testscr str expn_stu el_pct meal_pct calw_pct
```

```
*****列表报告结果*****
```

```
reg testscr str, r
```

```
esti store model1
```

```
reg testscr str el_pct, r //控制英语学习者百分比
```

```
esti store model2
```

```
reg testscr str el_pct meal_pct, r //控制英语学习者百分比和午餐补助学生百分比
```

```
esti store model3
```

```
reg testscr str el_pct calw_pct, r //控制英语学习者百分比和援助计划学生百分比
```

```
esti store model4
```

```
reg testscr str el_pct meal_pct calw_pct, r
```

```
esti store model5
```

```
esti table model*, stats(r2_a N) b (%5.2f) se (%5.3f)
```

/*总的结论：控制学生特征后，生师比对成绩的影响大约减少了一半，这个估计的影响对哪一个特定的控制变量被包含在回归中不是非常敏感的，在所有情况下，生师比的系数在 5%的水平下在统计上都是显著的。在含有控制变量的四个模型中（2）—（5），如果保持学生特征不变，平均每名教师减少 1 名学生，估计将使平均成绩提高约 1 分。

学生特征对于成绩是非常有用的预测因子，生师比仅解释了成绩变化的 0.049,当增加学生特征变量时，调整 R 方跳跃增加，在模型（3）中达到了 0.773。

个别地看，控制变量在统计上不总是显著的，在模型（5）中，享有收入援助的学生的非分比系数为零的假设在 5%的水平下没有被拒绝，因此在基准设定（3）的回归中，增加这个控制变量对估计系数及其标准误的影响可以忽略，所以至少对目前的分析而言，新增这个控制变量是多余的。

18.3 非线性回归分析

前面的模型设定中假设总体回归函数是线性的，实际上，在大班中减少一名学生对成绩造成的影响可能非常不同于对小班造成的影响（如班级规模过大，使得老师除了控制班级秩序外几乎不能做什么），如果是这样的话，总体回归线与生师比变量之间就不是简单的线性关系，而是关于生师比的非线性函数。我们还可能想到，降低哪些英语学习者百分比高地区的生师比，仍在学习英语的孩子更可能受益于较多的一对一的关注，因此生师比对成绩的影响依赖于第三个因素，即英语学习者的非分比。*/

*非线性回归分析

```

use caschool.dta, clear
**** 数据转化 ****
gen avginc2 = avginc*avginc
gen avginc3 = avginc2*avginc
gen loginc = ln(avginc)
gen logtest = ln(testscr)
gen loginc2 = loginc*loginc
gen loginc3 = loginc2*loginc
gen histr = (str>=20)
gen hiel = (el_pct >= 10)
gen hisxhie = histr*hiel
gen strxhie = str*hiel
gen strxelpc = str*el_pct
gen sttr2 = str*str
gen sttr3 = sttr2*str
gen str2hie = sttr2*hiel
gen str3hie = sttr3*hiel

reg testscr str el_pct meal_pct, r //线性回归模型，基准设定
esti store model1
reg testscr str el_pct meal_pct loginc, r //控制对数收入的影响
esti store model2
reg testscr str hiel strxhie, r //考虑生师比与英语学习者百分比的交互影响
esti store model3
test str strxhie //联合假设的 F 统计量和 P 值
reg testscr str hiel strxhie meal_pct loginc, r //控制学生特征的基础上考虑交互影响
esti store model4
test str strxhie //联合假设的 F 统计量和 P 值
reg testscr str sttr2 sttr3 hiel meal_pct loginc, r //考虑生师比的非线性影响
esti store model5
test str sttr2 sttr3 //联合假设的 F 统计量和 P 值
test sttr2 sttr3 //联合假设的 F 统计量和 P 值
reg testscr str sttr2 sttr3 hiel strxhie str2hie str3hie meal_pct loginc, r //考虑所有因素
esti store model6
test str sttr2 sttr3 strxhie str2hie str3hie //联合假设的 F 统计量和 P 值
test sttr2 sttr3 //联合假设的 F 统计量和 P 值
test strxhie str2hie str3hie //联合假设的 F 统计量和 P 值
reg testscr str sttr2 sttr3 el_pct meal_pct loginc, r //除交互影响外所有因素
esti store model7
test str sttr2 sttr3 //联合假设的 F 统计量和 P 值
test sttr2 sttr3 //联合假设的 F 统计量和 P 值
esti table model*, stats(r2_a N) b (%5.2f) se (%5.3f)

```

/*总的结论：设定（1）没有对收入进行控制，设定（2）将收入对数作为额外控制变量包括在回归中时，收入对数在 1% 的水平下在统计上显著，而且生师比的系数几乎接近 0，降到-0.73，尽管它在 1% 的水平下在统计上仍然是显著的。这表明，在回归设定中包括收入的对数作为遗漏变量偏差的一个控制因素是恰当的。

在设定（3）中，将含有英语学习者百分比高低的二元变量包括进来，并且与生师比交互，但不含经济控制变量，此时生师比单独地看并不显著。当经济控制变量被添加进来时，见模型（4），生师比的系数发生了变化，但在这两种情况下，交叉项的系数在 5% 的水平下都不显著。但所有含有生师比影响变量（包括交互项）的联合显著性检验表明，生师比与英语学习者对学

习成绩影响均为零的假设被拒绝了。

通过包括生师比的一个三次项，回归（5）检验了改变生师比所产生的效应是否依赖于生师比的值，检验这种关系是线性的零假设，在 1% 的显著性水平下被拒绝。

回归（6）进一步检验了生师比的效应，即是否存在不仅依赖于生师比的值，还依赖于英语学习者比率的现象，通过引入生师比与英语学习者二值虚拟变更的交互项，我们可以检验成绩与生师比之间的关系的总体回归函数，对高低不同的英语学习者的百分比来说是否不同，结果 F 统计量在 % 的水平下是显著的，但在 1% 的水平下不显著。

回归（7）是对回归（5）的修正式，它用连续变量代替了二元变量来控制地区内英语学习者的百分比，当进行这种修正时，其他回归因子的系数没有发生实质性的变化，这表明回归（5）中的结果对回归中实际所使用的测度英语学习者的百分比的指标不敏感。

在所有的回归设定中，生师比不进入回归的假设在 1% 的水平下都被拒绝。

18.4 回归模型的有效性

你相信自己的研究结论吗？你能让他人相信你的结论吗？是什么因素决定了回归模型是可靠的还是不可靠的？在什么时候计量模型能提供一个关于因果效应的有用估计？什么时候不能做到这一点。回答这些问题需要研究有效性的概念。包括内部有效性和外部有效性。

外部有效性

如果其推断和结论能够从所研究的总体和环境设定中，推广到其他总体和环境，那么这个分析就是外部有效的。比如在药物实验中，通常用小白鼠做实验，对小白鼠有效的药物对人也会同样有效吗？同样是健康安全的吗？前面对生师比和成绩的研究是在加州进行的，这个结论对其他的州，如马萨诸塞州也适用吗？这是外部有效性问题。那么，哪些因素会对外部有效性构成威胁呢？外部有效性也有两个方面，一是总体间的差异，二是环境设定间的差异。总体间的差异包括总体的特征不同（如老鼠与人），地理的差异（平原和山区），研究过时（改革开放前后）。环境设定间的差异包括机构环境差异（国企与民营企业），法律差异或自然环境差异。显然，总体和外部环境越接近，外部有效性越强。

如何评价一项研究的外部有效性呢？可以通过对比两项研究的结果进行检验，两个研究的相似结论支持了外部有效性的断言，而结论中不易解释的差异对外部有效性提出了质疑。

外部有效性的检验（加州和马萨诸塞州的对比）*/

*****马萨诸塞州的数据定义*****;

```
use mcas.dta, clear
* code          District Code (numerical)
* municipa      Municipality (name)
* district      District Name
* totsc4        4th grade score (math+english+science)
* totsc8        8th grade score (math+english+science)
* regday        Spending per pupil, regular
* specneed      Spending per pupil, special needs
* bilingual     Spending per pupil, bilingual
* occuppay      Spending per pupil, occupational
* tot_day       Spending per pupil, Total
* tchratio      Students per Teacher
* s_p_c         Students per Computer
* spec_ed       % Special Education Students
* lnh_pct       % Eligible for free/reduced price lunch
* avgsalry      Average Teacher Salary
* percap        Per Capita Income
* pctel         Percent English Learners
```


*****生成新变量名，使之与加州数据一致*****;

* rename variables -- same as CA dataset

gen str = tchratio

gen testscr=totsc4

gen el_pct = pctel

gen avginc = percap

gen meal_pct = lnch_pct

gen avginc2 = avginc*avginc

gen avginc3 = avginc2*avginc

gen loginc = ln(avginc)

gen hiel = (el_pct > 0)

gen strxhiel = str*hiel

gen sttr2 = str*str

gen sttr3 = sttr2*str

****描述性统计分析*****;

sum testscr str el_pct meal_pct avginc

sum el_pct, d

****回归分析，表 4*****;

reg testscr str, r

esti store m1

reg testscr str el_pct meal_pct loginc, r

esti store m2

reg testscr str el_pct meal_pct avginc avginc2 avginc3, r

esti store m3

test avginc2 avginc3

reg testscr str sttr2 sttr3 el_pct meal_pct avginc avginc2 avginc3, r

esti store m4

test str sttr2 sttr3

test sttr2 sttr3

test avginc2 avginc3

reg testscr str hiel strxhiel meal_pct avginc avginc2 avginc3, r

esti store m5

test str strxhiel

test avginc2 avginc3

test hiel strxhiel

reg testscr str meal_pct avginc avginc2 avginc3, r

esti store m6

test avginc2 avginc3

esti table m*, stats(r2_a N) b (%5.2f) se (%5.3f)

/*外部有效性的结论：两州在总的来看比较相似，差异主要有：马州的平均成绩高而加州的平均成绩低，直接比较没有意义（但可以标准化以后进行比较，方法是对成绩分别进行标准化变换），两州的平均生师比也有差异，加州约高 2 个单位。马州的英语学习者百分比比较低（1.1%）而加州高（15.8%），享有午餐补助的学生百分比方面马州也比加州低很多。在收入方面，加州的平均收入低且变异更大。另外加州的观察值较多（420）而马州只有 220 个观察值。

由于这些特点，以第（3）个模型为基准，两州的估计结果有三点类似：

- （1） 添加控制学生背景的变量后，生师比系数降低了 68%，大约在-0.7 与-0.6 之间。
- （2） 加州生师比的真实系数为零的假设在 1%的显著性水平下被拒绝，而马州在 5%的水平上被拒绝显著，这种差异主要是因为样本容量不同。
- （3） 减少生师比的效应并不十分依赖于该地区英语学习者的百分比。

不同之处：证据表明加州的学习成绩与生师比的关系是非线性的，而马州没有明显的非线性关系。

综合结论：通过将成绩标准化之后分别回归，得到加州生师比减少 2 个单位将使考试成绩的标准差变动 0.076，马州减少 2 个单位将使成绩标准差变动 0.085。因此预测 2 个单位的幅度削减生师比，会使成绩大约提高 0.08 个地区间考试成绩分布的标准差。这个效应在统计上是显著的，但是它非常小。

内部有效性：

如果一项关于因果效应的统计推断对所研究的总体来说是有效的，那么该统计分析就是内部有效的。内部有效性有两个组成部分，（1）因果效应的估计量是无偏的且一致的。（2）假设检验应该具有希望达到的显著性水平，置信区间应该在重复抽样时以 95% 的概率包含真实的总体因果效应。那么，在什么条件下，能使得估计的因果效应满足这些要求呢？这涉及到 OLS 估计的假设和定理。

导致 OLS 估计量可能有偏主要有五个原因：遗漏变量、回归函数形式误设、自变量测量误差、样本选择、联立因果关系。这五个方面的问题导致自变量与误差项相关，从而零条件均值假设不再成立。如何处理这五个方面的问题呢？

（1）遗漏变量偏差：当这些被遗漏变量可得时的处理方法 A 根据研究目标和问题明确确定关键系数，比如生师比。B 使用理论和经验，建立基准模型并识别潜在的遗漏变量。C 将识别出来的可疑变量引入模型进行检验，如果这些变量是统计上显著的，或者引入后导致关键系数发生明显变化，则应该保留这些变量。D 完全披露，列表报告不同的模型设定及估计结果。

当遗漏变量观测不到时的解决办法 A 使用面板数据 B 使用工具变量 C 做随机化的实验或利用自然实验机会收集数据。

（2）函数形式误设偏误：考虑自变量的高次项或对数形式。然后进行模型设定的敏感性检验。

（3）变量误差：A 工具变量回归，找到一个与真实值 X 相关，但与测量误差不相关的变量做为工具变量 B 导出测量误差的数学模型，对估计结果进行修正。

（4）样本选择偏差：用样本选择模型 heckman。

（5）联立因果关系偏误：如果政府计划降低师生比，而且根据学习成绩来分配教师（在成绩差的地方分配更多教师），则成绩为因，生师比为果，反向因果关系出现将导致联立性。解决的办法是：A 工具变量回归 B 随机化控制实验或自然实验。

导致 OLS 标准误不一致的主要有两个原因：（1）对异方差的不恰当处理 （2）观测值之间误差项的相关性。对异方差的处理主要是运用异方差稳健性标准误和相应的统计量，对误差项之间的相关性要采用 HAC 标准误。

在班级规模和学生成绩的案例中，遗漏变量可能存在，例如生师比可能与老师质量有关（若较好的老师被吸引到生师比较小的学校），而且教师质量也会影响考试成绩。在函数形式误设方面，主要结论对使用不同的非线性回归设定并不是很敏感。变量误差主要来自于地区收入，该数据是 1990 年的普查数据，如果目前 1998/1999 的经济结构发生了巨大变量，那么对该变量的测度就是不精确的。由于所有的学区都被选择了，因此不存在样本选择偏差。由于对学校的资源分配并非以成绩为标准的，因此，不存在联立因果偏误。

在前面的研究中，都采用了异方差稳健的标准误，因此，异方差问题不会威胁到内部有效性，但是因为样本由州内所有的学区组成，因此观测值之间误差项可能相关并威胁到标准误的一致性。这些内容需要在更高级的课程中学习。

教育主管现在能使用该项研究来帮助决定是否要减小班级规模。需要权衡政策调整的成本和收益，增加教师和教室需要一大笔费用，带来的利益是学习成绩提高了。可能还有其他的利

益，如降低辍学率和提高未来的收入，这需要另外的研究。

18.5 实验与自然实验

20 世纪 80 年代后期，田纳西州执行了一个很大的、花费 1200 万美元的随机化控制实验 (Student-Teacher Achievement, STAR)，以确定减小班级规模是否是一种改善初等教育的有效方法。这个实验的结论已强烈地影响到我们对减小班级规模效应的理解。

STAR 项目在 1985-1986 学年初将参与该实验的学校的学生随机地分配到三种类型的班级中，老师也被随机地分配到三种类型的班级。这三种班级是：常规班，22—25 个学生，一位老师。小型班，13—17 个学生，一位老师。常规助教班，常规班，但配有一位教师和一位助教。

最初的实验设计是学生们要在他们所分配的班级中呆四年，从幼儿园到小学三年级，但因家长的抱怨，在一年级时，最初被分配到常规班的学生被随机地重新分配到有助教的常规班和没有助教的常规班，最初被分配到小班的学生仍留在小班。在项目的第一年，大约有 6400 名学生参加了 108 个小班，101 个常规班和 99 个有助教的常规班，在整个研究的 4 年间，共有 80 所学校的约 11600 名学生参加了该项研究。

实验协议并没完全遵行，约有 10% 的学生换了班。由于家长压力导致的换班可能会对实验结论产生影响，另外一种偏离是，学生迁入和迁出导致班级规模随着时间而发生了变化。

通过比较三个班级的平均成绩，即可以获得结论，用回归模型来表达，则为：

$$\text{成绩} = \beta_0 + \beta_1 \text{smallclass} + \beta_2 \text{regaide} + u$$

其中 tscorek 为标准化考试成绩，sck 为是否小班，参照为常规班。rak 为有无助教，参照组也是常规班。*/

*****环境设定*****

```
use star_sw.dta, clear
```

*****差分估计值*****

```
*-----幼儿园-----
```

```
sum tscorek sck rak if stark==1
reg tscorek sck rak if stark==1, r
```

```
*-----一年级-----
```

```
sum tscore1 sc1 ra1 if star1==1
reg tscore1 sc1 ra1 if star1==1, r
```

```
*-----二年级-----
```

```
sum tscore2 sc2 ra2 if star2==1
reg tscore2 sc2 ra2 if star2==1, r
```

```
*-----三年级-----
```

```
sum tscore3 sc3 ra3 if star3==1
reg tscore3 sc3 ra3 if star3==1, r
```

/*结论：减小小班规模对考试成绩有影响，但是给常规班增加一个助教对考试成绩有比较小的影响，可能是零。

对幼儿园而言，相对于在常规班，在小班中考试成绩的效应是增加 13.9 分，在有助教的常规班对考试成绩的估计效应是增加 0.31 分。对每个年级而言，小班没有提高成绩的零假设在 1% 的显著性水平下被拒绝。但是除了一年级，有助教的常规班没有提高成绩的零假设不能被拒绝。除了一年级，幼儿园、二年级和三年级中，小班所估计的成绩提高幅度相似。*/

*****对幼儿园带有额外自变量的差分估计量*****

```
preserve
keep if stark==1
reg tscorek sck rak, r
est store m1
```

```

reg tscorek sck rak totexpk, r //增加教师的工作年限 totexpk 为额外自变量
est store m2
areg tscorek sck rak totexpk, r absorb(schidkn) //增加学校固定效应
esti store m3
*增加性别 boy、免费午餐 freelunk, 黑人 black, 种族 other, 学校等额外控制变量
areg tscorek sck rak totexpk boy freelunk black other, r absorb(schidkn)
esti store m4
esti table m*, stats(r2_a N) b (%5.2f) se (%5.3f)
table sck,c(m totexpk m boy m freelunk m black m other)
log close

```

*=====end=====

/*结论：增加额外的控制变量并没有改变不同处理的估计因果效应。但这些额外的自变量增加了回归的调整 R 方，而且使班级规模效应的标准误从 2.45 减小到 2.16。*/

截取与断尾

断尾

```
clear
matrix m=(1,2,3,4)
drawnorm x1-x4,n(1000) m(m) //条件分布
gen n=_n
reshape long x,i(n) j(m)
gen y=x
replace y=. if y>=4 //大于4的样本被断尾
gen z=normalden(y,m,1)*(1-normal(y-m)) //纠正办法
g f=normalden(x,m,1)
tw (line f z x,sort) (kdensity y),by(m) xline(4)
```

截取

```
clear
matrix m=(1,2,3,4)
drawnorm x1-x4,n(1000) m(m)
gen n=_n
reshape long x,i(n) j(m)
gen y=x
replace y=4 if y>=4 //大于4的样本被截取
gen z=normalden(y,m,1)*(1-normal(y-m))
g f=normalden(x,m,1)
tw (line f z x,sort) (kdensity y),by(m) xline(4)
```

参考文献