《计量软件在金融中应用》

案例集

沈根祥

（上海财经大学经济学院）

2021.10

案例 1：P2P 网贷交易数据匹配

案例背景：

P2P 是一种人到人（peer-to-peer）融资方式的简称。为解决小为企业的融资问题，P2P 融资方式在我国得到了快速发展和野蛮增长，涌现出各式各样的 P2P 网贷平台，在便利中小企业融资的同时，增加了金融市场风险。研究 P2P 网贷平台融资主体的交易行为，能够有效监测和预防金融风险。

案例描述：—— 提出问题

本案例采用 SAS 软件对 P2P 网贷数据进行处理：在 100 万条网贷时间序列数据中，要求统计同一个债权卖出者在其卖出日期后的 5 天、7 天、10 天、15 天及 30 天内是否有再买债权以及购买次数，以及再买债权的种类。

采用 proc import 过程将数据集导入 SAS 数据 p2p 后，数据集具有如下结构（前 10 条记录）：

	amount	sellid	buyid	id	Time
1	93263.03000000013	1000023	1144058	292173	07NOV16:22:21:43
2	98536.29000000008	1000023	477211	292178	27OCT15:17:04:08
3	98536.29000000008	1000023	1569350	292178	27OCT15:17:04:05
4	28848.85000000001	1000023	318100	289648	07APR15:19:27:33
5	17779.79	1000023	350832	289652	07APR15:19:28:01
6	28848.85000000001	1000023	560400	289648	07APR15:19:27:45
7	93263.03000000013	1000023	259210	292173	07NOV16:22:21:08
8	17779.79	1000023	1466757	289652	07APR15:19:28:18
9	46034.17999999998	1000023	10626	292213	07NOV16:21:57:28
10	44511.51999999994	1000023	259210	292221	07NOV16:21:58:25

数据集中变量的含义为
id：债权种类
sellid：债权卖出者身份
buyid：债权买入者身份
time：债权交易时间
amount：债权交易量

问题归类：

用 SAS 进行大型数据集查询

教学过程：

step1：如何进行？

采用 data 步中的多 set 语句进行查询。

首先用 data 步生成数据集 work.p2p，其中变量 date 为债券交易日期，并按日期进行排序。

```
data p2p;
set book.p2p;
date=datepart(time);
time=timepart(time);
```

```
format date yymmdd10.;
format time time.;
keep date time sellid buyid id;
run;
proc sort data=p2p;
by date;
run;
```

为避免程序数据向量 PDV 中变量值覆盖，要用 rename 语句对其中之一 set 语句读入 PDV 中的变量进行改名。程序代码如下

```
data buy;
set p2p(rename=(sellid=sellid1 buyid=buyid1 date=date1))
nobs=gs curobs=cn;
date=0;
do i=cn+1 to gs while(date<date1+30);
set p2p point=i;
if sellid=buyid1 then output;
end;
run;
```

step2：什么问题?

运行程序发现，查询 100 万条记录用时太长！尤其是查询后 30 天是否有再买债权时用时更长，几乎达到不可行的地步。

step3：产生原因

基于 data 的双 set 语句查询计算量巨大！以查询后 30 天是否有再买债权为例。对 100 万条中的每条记录，需要查询其后 30 天的交易记录进行比较，由于数据库中一天交易记录很多，需要查询的记录也会很多。图示为

	sellid	buyid	id	Time	date
1	100545	232358	91393	11:43:45	2013-11-27
2	100545	268550	91393	11:43:53	2013-11-27
3	100545	445616	33439	11:43:42	2013-11-27
4	100666	219643	62130	13:12:26	2013-12-04
5	100666	132749	70762	13:11:56	2013-12-04

2	100545	268550	91393	11:43:53	2013-11-27
3	100545	445616	33439	11:43:42	2013-11-27
4	100666	219643	62130	13:12:26	2013-12-04
5	100666	132749	70762	13:11:56	2013-12-04
6	100666	209093	62130	13:12:23	2013-12-04
7	100666	413851	47024	13:12:55	2013-12-04
8	100666	510906	62130	13:12:58	2013-12-04
9	100666	209093	62130	13:12:52	2013-12-04
10	100666	400452	70762	13:11:46	2013-12-04
11	100666	385430	72487	9:35:11	2013-12-04
12	100666	432817	62130	13:12:41	2013-12-04
13	100666	82784	72487	9:37:10	2013-12-04
14	100666	334331	47024	13:12:45	2013-12-04
15	100666	461422	72487	9:35:18	2013-12-04
16	100666	226389	72487	9:37:21	2013-12-04
17	100666	448886	47024	13:12:36	2013-12-04
18	100666	132749	70762	13:12:14	2013-12-04
19	100666	448868	70762	13:11:40	2013-12-04
20	100666	433480	62130	13:13:10	2013-12-04
21	100666	234726	72487	9:34:37	2013-12-04
22	100666	209093	70762	13:11:42	2013-12-04
23	100666	209093	62130	13:12:40	2013-12-04
24	100666	513844	62130	13:12:46	2013-12-04
25	100666	511645	72487	9:37:16	2013-12-04
26	100666	500887	72487	9:37:07	2013-12-04
27	100666	193417	47024	13:12:52	2013-12-04
28	100666	193214	72487	9:37:54	2013-12-04
29	100498	413431	63370	15:37:17	2013-12-06
30	100498	395278	64119	15:36:40	2013-12-06
31	100498	233762	63368	15:37:30	2013-12-06
32	100498	239292	64432	15:36:05	2013-12-06

如果按照每条记录后 30 天有 1000 条记录，则查询次数为

$$10^6 \times 10^3 = 10^9 = 10 \text{ 亿}$$

step4：解决方案

如果采用 SQL 或者 Hash 表进行查询，会很快完成。在只限于 SAS/Base data 步时，可采用分割数据集的方法，分数据集进行查询，尽管不能节省总体查询时间，但不至于造成宕机，解决了问题的可行性。

首先需要将数据集分割为小的数据集。将 100 万条记录的数据集分割为 10 万条记录的 10 个小的数据集，对每个小的数据集实施查询，最后把查询结果进行合并。

1）分割数据集

采用 data 步分割数据集较为简单，当分割个数较少时可以采用多个 if-then-output-datasetn 语句。例如分割为 10 个数据集的代码为（为节省篇幅，仅列出 5 个）

```
data p2p1 p2p2 p2p3 p2p4 p2p5;
set p2p nobs=total;
nob=int(total/10);
inter=100;
if _n_<=nob then output p2p1;
if nob-inter=<_n_<=2*nob then output p2p2;
if 2*nob-inter=<_n_<=3*nob then output p2p3;
if 3*nob-inter=<_n_<=4*nob then output p2p4;
if 4*nob-inter=<_n_<=5*nob then output p2p5;
run;
```

要推广这种操作到一般情形，需要借助宏和宏循环。

首先定义生成数据集名的宏

```
%macro split(n,overlap);
%do i=1 %to &n;
data p2p&i;
set p2p nobs=total;
nob=int(total/&n);
tem=&i;
if nob*tem-&overlap=<_n_<=nob*(tem+1) then output p2p&i;
run;
%end;
%mend;
%split(10,100);
```

2）分数据集查询

由于分出的数据集较多，为避免程序代码重复，可采用两种方法设计程序。

i)采用宏

```
%macro match(n);
%do i=1 %to &n;
data buy&i;
set p2p&i(rename=(sellid=sellid1 buyid=buyid1 date=date1))
nobs=gs curobs=cn;
```

```
date=0;
do i=cn+1 to gs while(date<date1+30);
set p2p point=i;
if sellid=buyid1 then output;
end;
run;
%end;
%mend;
%match(10);
```

ii)采用 data 步中的 call execute( )

采用宏进行分数据集查询需要掌握宏语言。如果不具备宏语言基础，则可以通过调用 data 中的例程 execute( )达到同样效果。

首先生成数据集名变量数据集 name

```
data name;
do i='1' to '10';
name1=cats('p2p',i);
name2=cats('buy',i);
output;
end;
keep name1 name2;
run;
```

生成的数据集 name 中变量 name1 为查询数据集（输入数据集）名，name2 查询结果数据集（输出数据集）名:

	name1	name2
1	p2p1	buy1
2	p2p2	buy2
3	p2p3	buy3
4	p2p4	buy4
5	p2p5	buy5
6	p2p6	buy6
7	p2p7	buy7
8	p2p8	buy8
9	p2p9	buy9
10	p2p10	buy10

然后在 data 步中调用子程序 execute( )

```
data _null_;
set name;
txt1="data "||name2||" ;";
txt2="set "||name1||"(rename=(sellid=sellid1 buyid=buyid1
date=date1)) nobs=gs curobs=cn;";
txt3="date=0;do i=cn+1 to gs while(date<date1+30);";
txt4="set p2p point=i;if sellid=buyid1 then output;end;";
txt=txt1||txt2||txt3||txt4;
call execute(txt);
run;
```

需要注意，由于 cats 函数连接的字符总数收到限制，程序中采用字符连接运算符“||“而不是函数 cats( )进行字符连接。此外，函数 cats( )会删除所连接字符的尾部空格，导致语句中关键词之间的空格被删除而发生错误。

总结讨论:

问题分解：当需要处理的数据巨大而不方便处理时，采用数据集分割的方法提出可行的处理方法。在大数据情况下，很多问题都可以采用类似的分解方法来完成。如果能够采用并行计算，则不仅使问题可行，还能提高处理效率。

不同方法：SAS 编程语言十分灵活，对同一问题往往由不同的实现方法。在本案例中，采用宏语言能够实现的操作，也可以使用最为基本的 data 步结合例程调用 call execute( )来实现，使得不熟悉宏语言的同学也能完成操作。

思考题:

在采用 data 步调用例程 execute( )中，如果不实现生成包含数据集名字的变量 name 的数据集 name，而在 data 内用 do-end 循环是否能够完成响应的任务？（提示：用 do-end 循环，循环指标变量 i 的值取字符“1“到”10“，然后用前缀”p2p“和变量 i 组合得出输入数据集名文本，用前缀”buy“和变量 i 组合得出输出输名文本）。

案例 2：国债收益率曲线构建

案例背景：

利率期限结构是指不同到期时间的无风险零息债券到期收益率与到期期限之间的函数关系，对应的图形称为收益率曲线。利率期限结构反映了投资者对利率未来走向的预期，具有丰富的信息含量，是投资决策的重要参考。中央银行通过公开市场操作影响利率期限结构，熨平周期性因素和偶然因素带来的期限结构扭曲，引导市场对未来形成正确预期。利率期限结构模型分为静态模型和动态模型。静态模型的研究重点在于用市场交易债券的到期收益率尽可能合理拟合整条收益率曲线，得出所有期限债券的收益率。静态模型常采用样条函数构造收益率曲线，在拟合不同形状收益率曲线时有足够的灵活性，但样条函数模型得出的远端收益率常常为负值，并且模型参数过多。Nelson 和 Siegel（1987）采用拉盖尔函数（Lagueere function）构造出四个参数的利率期限结构模型，称为 Nelson-Siegel 模型（以下简称 NS 模型）[1]。NS 模型以简洁性和拟合收益率曲线形状的灵活性得到广泛应用，很多国家的中央银行采用 NS 模型构建和发布收益率曲线，中国外汇交易中心从 2005 年 6 月开始采用 NS 模型构造和发布我国银行间市场国债交易收益率曲线。

静态模型能够准确拟合单个时点上的整条收益率曲线，却不能刻画收益率曲线随时间变化的动态规律，不能对收益率进行预测。为研究债券收益率动态变化以及与其他经济变量的关系，人们提出了各种动态利率期限结构模型，包括 Vasicek 模型、CIR 模型和仿射模型。借助于 NS 模型在静态收益率曲线拟合上的良好表现，Diebold 和 Li 将 NS 模型推广到动态模型，提出了动态 Nelson-Siegel 模型[2]（以下简称 DNS 模型）。DNS 模型将债券到期收益率表示为水平、斜率和曲率三个因子的线性函数，将因子载荷约束为 NS 模型的形式。DNS 模型十分简练，模型中的因子具有明确的经济意义。

实际经济活动中的短期借贷流动性远远高于长期借贷，债券交易中的短期债券交易活跃而长期债券交易较为清淡，收益率曲线短端波动剧烈而长端平缓，收益率曲线的近端包含更为丰富的经济信息，同时投资者也更为关注短期利率的变化。基于此，利率期限结构模型应该具备对短期利率变动更为灵活的刻画和捕捉能力。DNS模型用水平（level）、斜率（slope）和曲度（curvature）三个因子刻画收益率曲线的形状（简称为LSC模型），用因子的一阶自回归模型描述收益率曲线的动态变化，达到较好的静态拟合和动态预测效果。

案例描述：—— 提出问题

我国债券市场5年期以下的中短期债券交易最为活跃，本案例在DNS模型的基础上，引入第二个斜率因子，构造双斜率四因子动态利率期限结构LSSC模型，以增强模型对我国收益率曲线近端的静态拟合和动态预测，以期对债权实现更为客观的定价。

数据呈现和模型估计采用SAS/ETS中的过程Proc SSM实现。

1. 我国的国债收益率时间序列

我国债券市场分为银行间市场与交易所市场，银行间市场交易量大，交易活跃度高，比交易所市场更具代表性。本文采用银行间中债即期数据，数据来自 wind 金融数据库，选取 2006 年 3 月至 2013 年 11 月的周数据。这一时期中国债券市场波动较大，经历了牛熊交替，具有代表性：2006 年至 2007 年的宽松货币政策使得债券市场相对繁荣，2007 年至 2008 年经济过热，央行从紧的货币政策推高国债收益率，2008 年金融危机后，央行实施宽松货币政策，债券市场处于阶段性繁荣。债券期限选取 6、12、18 和 24 个月的短期期限，30、36、42、48、55、60、66、72、78、84、90、96、102、108、114、120 个月的中长期期限和、144、168 的长期期限。图一不同期限债券收益率序列三维图。

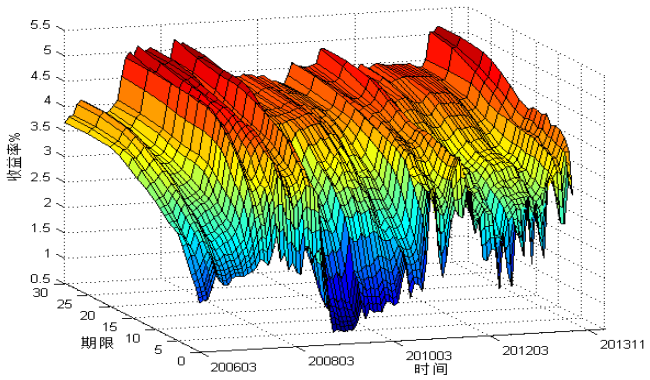


图 1 银行间市场债券收益率时序图

从图中看出，不同期限债券的收益率变化具有很强的协同性，表明受到共同因素的影响。不同期限收益率变动呈现出鲜明特点：短期收益率波动剧烈，长期收益率波动缓和，与经典的利率期限结构理论一致。

2. 四因子动态利率期限LSSC模型

静态Nelson-Siegel模型广泛用于利率期限结构的静态拟合，具体形式为：

$$y_t(\tau)=\beta_1+\frac{1-e^{-\lambda\tau}}{\lambda\tau}\beta_2+\left[\frac{1-e^{-\lambda\tau}}{\lambda\tau}-e^{-\lambda\tau}\right]\beta_3 \qquad (1)$$

其中 $y_t(\tau)$ 表示期限为 τ 的债券在时刻 t 的到期收益率，λ 、β_1、β_2 、β_3 为参数。通过参数取值变化，NS模型具有向上、向下、驼峰、倒驼峰四种形状，对应了收益率曲线四种典型形状，对利率期限结构具有很好的拟合效果。从（1）得出

$$y_t(\infty)=\beta_1,\quad y_t(\infty)-y_t(0)=-\beta_2$$

表明参数 β_1 和 β_2 具有明确的经济意义：β_1 为长期利率（无限期债券收益率），β_2 为长短期利差。参数 β_3 没有明确的经济意义，但其对应项 $\beta_3[(1-e^{-\lambda\tau})/\lambda\tau-e^{-\lambda\tau}]$ 对于描述收益率曲线的中间部分十分重要，λ 称为衰减参数，决定了随着期限 τ 增加，因子载荷项中的 $e^{-\lambda\tau}$ 衰减到0的速度。

为克服静态模型NS模型无法描述收益率曲线随的动态演化过程，也不能进行预测的缺陷。Diebold和Li将NS模型（1）中的回归系数 β_1 、β_2 和 β_3 推广为时变变量，并根据其实际经济含义命名为水平因子、斜率因子和曲度因子，分别用 L_t 、S_t 和 C_t 表示，将对应的1、$(1-e^{-\lambda\tau})/\lambda\tau$ 和 $(1-e^{-\lambda\tau})/\lambda\tau-e^{-\lambda\tau}$ 看作因子载荷，用一阶向量自回归描述因子的动态变化，得出动态NS模型（以下简称DNS模型）[3]:

$$y_t(\tau)=L_t+\frac{1-e^{-\lambda\tau}}{\lambda\tau}S_t+\left[\frac{1-e^{-\lambda t}}{\lambda\tau}-e^{-\lambda\tau}\right]C_t$$

$$\begin{bmatrix}L_t-\mu_L\\S_t-\mu_S\\C_t-\mu_C\end{bmatrix}=A\begin{bmatrix}L_{t-1}-\mu_L\\S_{t-1}-\mu_S\\C_{t-1}-\mu_C\end{bmatrix}+\begin{bmatrix}\eta_{Lt}\\\eta_{St}\\\eta_{Ct}\end{bmatrix} \qquad (2)$$

，μ_L 、μ_S 和 μ_C 为常数自回归模型均值，A 为自回归系数矩阵，η_t 为误差向量。

设市场上有 N 种期限的债券，对应期限为 $\tau_1,\tau_2,\cdots,\tau_N$ ，考虑到实际数据中包含有测

量误差，$y_t(\tau)$中包含随机误差项，DNS模型可表示为状态空间模型

$$\begin{bmatrix} y_t(\tau_1) \\ y_t(\tau_2) \\ \cdots \\ y_t(\tau_N) \end{bmatrix} = \begin{bmatrix} 1 & \frac{1-e^{-\lambda\tau_1}}{\lambda\tau_1} & \frac{1-e^{-\lambda\tau_1}}{\lambda\tau_1} - e^{-\lambda\tau_1} \\ 1 & \frac{1-e^{-\lambda\tau_2}}{\lambda\tau_2} & \frac{1-e^{-\lambda\tau_2}}{\lambda\tau_2} - e^{-\lambda\tau_2} \\ \cdots & \cdots & \cdots \\ 1 & \frac{1-e^{-\lambda\tau_N}}{\lambda\tau_N} & \frac{1-e^{-\lambda\tau_N}}{\lambda\tau_N} - e^{-\lambda\tau_N} \end{bmatrix} \begin{bmatrix} L_t \\ S_t \\ C_t \end{bmatrix} + \begin{bmatrix} \varepsilon_t(\tau_1) \\ \varepsilon_t(\tau_2) \\ \cdots \\ \varepsilon_t(\tau_N) \end{bmatrix} \tag{3}$$

$$\begin{bmatrix} L_t - \mu_L \\ S_t - \mu_S \\ C_t - \mu_C \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} + A \begin{bmatrix} L_{t-1} - \mu_L \\ S_{t-1} - \mu_S \\ C_{t-1} - \mu_C \end{bmatrix} + \begin{bmatrix} \eta_{Lt} \\ \eta_{St} \\ \eta_{Ct} \end{bmatrix}$$

，观测方程误差项$\varepsilon_t \equiv [\varepsilon_t(\tau_1), \cdots, \varepsilon_t(\tau_N)]' \sim IIN(\mathbf{0}, diag(\sigma_1^2, \cdots \sigma_N^2))$为高斯白噪声向量序列，协方差矩阵为对角阵，$\sigma_i^2$为误差项$\varepsilon_t(\tau_i)$的方差。转移方程误差项$\eta_t \equiv [\eta_{Lt}, \eta_{St}, \eta_{Ct}]' \sim IIN(\mathbf{0}, H)$为高斯白噪声向量序列，协方差矩阵为$H$。观测方差误差项和因子不相关，和转移方程误差项也不相关，即

$$\begin{aligned} &\varepsilon_t \sim N(0, diag(\sigma_1^2, \sigma_2^2, \cdots, \sigma_N^2)), \quad \eta_t \sim N(0, H) \\ &E(\varepsilon_t L_t) = E(\varepsilon_t S_t) = E(\varepsilon_t C_t) = 0, \quad E(\varepsilon_t \eta_t') = 0 \end{aligned} \tag{4}$$

DNS模型中水平因子L_t用来刻画收益率曲线的整体变化，斜率因子则捕捉了收益率曲线近端的动态变化

我国债券市场长期（10年以上）债券发行量较少，交易不活跃，短期（1年期以下）债券的发行集中在6个月以上，中期债券交易集中，5年期以下交易最为活跃。鉴于此，本文对DNS模型进行扩展，添加第二个斜率因子，增强模型对收益率曲线近端动态变化规律的捕捉能力，得出双斜率因子LSSC四因子DNS模型：

$$\begin{aligned} &y_t(\tau) = L_t + \frac{1-e^{-\lambda_1\tau}}{\lambda_1\tau} S_t^{(1)} + \frac{1-e^{-\lambda_2\tau}}{\lambda_2\tau} S_t^{(2)} + \left[\frac{1-e^{-\lambda_1\tau}}{\lambda_1\tau} - e^{-\lambda_1\tau} \right] C_t \\ &\begin{bmatrix} L_t - \mu_L \\ S_t^{(1)} - \mu_s^{(1)} \\ S_t^{(2)} - \mu_s^{(2)} \\ C_t - \mu_C \end{bmatrix} = A \begin{bmatrix} L_{t-1} - \mu_L \\ S_{t-1}^{(1)} - \mu_s^{(1)} \\ S_{t-1}^{(2)} - \mu_s^{(2)} \\ C_{t-1} - \mu_C \end{bmatrix} + \begin{bmatrix} \eta_{Lt} \\ \eta_{St}^{(1)} \\ \eta_{St}^{(2)} \\ \eta_{Ct} \end{bmatrix} \end{aligned} \tag{5}$$

对于LSSC模型，给出如下两点说明：

（1）LSSC模型中，两个斜率因子的载荷具有相同的形式，但其中的衰减参数取不同值λ_1和λ_2。如果$\lambda_1 = \lambda_2$，$S_t^{(1)}$和$S_t^{(2)}$合并为一个因子$S = S_t^{(1)} + S_t^{(2)}$，四因子LSSC模型简化为DNS模型。因此，LSSC模型嵌套了DNS模型，是DNS模型的推广。

（2）在LSSC模型中，

$$\begin{aligned} &y_t(\infty) = L_t, \\ &y_t(0) = \lim_{\tau \to 0} y_t(\tau) = L_t + S_t^{(1)} + S_t^{(2)} \\ &y_t(\infty) - y_t(0) = -(S_t^{(1)} + S_t^{(2)}) \end{aligned}$$

，因此L_t仍然表示利率的整体水平，而利差则由两个斜率因子之和确定。将斜率因子载荷

关于期限 τ 在0附件二阶Taylor展开得出

$$y_t(\tau) = L_t + S_t^{(1)} + S_t^{(2)} - 0.5(\lambda_1 S_t^{(1)} + \lambda_2 S_t^{(2)})\tau + o(\tau)$$

，LSCC用两个斜率因子 $S_t^{(1)}$ 和 $S_t^{(2)}$ 的线性组合 $\lambda_1 S_t^{(1)} + \lambda_2 S_t^{(2)}$ 形成期限 τ 的线性函数，当 $S_t^{(1)}$ 和 $S_t^{(2)}$ 按照一阶向量自回归的转移方程变化时，模型能够更为灵活地刻画收益率曲线近端的形状和动态规律性。

（3）本文对DNS模型扩展的目的是增强模型对收益率曲线近端的描述能力，只在DNS模型上添加斜率因子 $S^{(2)}$ 。

如果债券期限为 $\tau_1, \tau_2, \cdots, \tau_N$ ，考虑到实际数据中包含有测量误差，在 $y_t(\tau)$ 中添加随机误差项 $\varepsilon_t(\tau)$ ，LSCC模型可表示为状态空间模型

$$\begin{bmatrix} y_t(\tau_1) \\ y_t(\tau_2) \\ \cdots \\ y_t(\tau_N) \end{bmatrix} = \begin{bmatrix} 1 & \frac{1-e^{-\lambda_1\tau_1}}{\lambda_1\tau_1} & \frac{1-e^{-\lambda_2\tau_1}}{\lambda_2\tau_1} & \frac{1-e^{-\lambda_1\tau_1}}{\lambda_1\tau_1} - e^{-\lambda_2\tau_1} \\ 1 & \frac{1-e^{-\lambda_1\tau_2}}{\lambda_1\tau_2} & \frac{1-e^{-\lambda_2\tau_2}}{\lambda_2\tau_2} & \frac{1-e^{-\lambda_1\tau_2}}{\lambda_1\tau_2} - e^{-\lambda_2\tau_2} \\ \cdots & \cdots & \cdots & \cdots \\ 1 & \frac{1-e^{-\lambda_1\tau_N}}{\lambda_1\tau_N} & \frac{1-e^{-\lambda_2\tau_N}}{\lambda_2\tau_N} & \frac{1-e^{-\lambda_1\tau_N}}{\lambda_1\tau_N} - e^{-\lambda_2\tau_N} \end{bmatrix} \begin{bmatrix} L_t \\ S_t^{(1)} \\ S_t^{(2)} \\ C_t \end{bmatrix} + \begin{bmatrix} \varepsilon_t(\tau_1) \\ \varepsilon_t(\tau_2) \\ \cdots \\ \varepsilon_t(\tau_N) \end{bmatrix}$$

$$\begin{bmatrix} L_t - \mu_L \\ S_t^{(1)} - \mu_s^{(1)} \\ S_t^{(2)} - \mu_s^{(2)} \\ C_t - \mu_C \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix} \begin{bmatrix} L_{t-1} - \mu_L \\ S_{t-1}^{(1)} - \mu_s^{(1)} \\ S_{t-1}^{(2)} - \mu_s^{(2)} \\ C_{t-1} - \mu_C \end{bmatrix} + \begin{bmatrix} \eta_{Lt} \\ \eta_{St}^{(1)} \\ \eta_{St}^{(2)} \\ \eta_{Ct} \end{bmatrix} \tag{6}$$

$$\varepsilon_t \sim N(0, diag(\sigma_1^2, \sigma_2^2, \cdots, \sigma_N^2)), \quad \eta_t \equiv (\eta_{Lt}, \eta_{St}^{(1)}, \eta_{St}^{(2)}, \eta_{Ct}) \sim N(0, H)$$

$$E(\varepsilon_t L_t) = E(\varepsilon_t S_t^{(1)}) = E(\varepsilon_t S_t^{(2)}) = E(\varepsilon_t C_t) = 0, \quad E(\varepsilon_t \eta_t') = 0$$

其中 ε_t 和 η_t 分别表示观测方程残差和转移方程残差。LSSC模型的转移方程为四维VAR(1)模型，误差项 η_t 为四维高斯白噪声，协方差矩阵 H 为4×4正定矩阵。

LSSC模型具有线性状态空间模型的形式，可以采用卡尔曼率算法计算样本对数似然函数，采用极大似然估计方法估计模型参数。将（6）式用矩阵符号表示为

$$y_t = \Gamma(\lambda_1, \lambda_2) F_t + \varepsilon_t$$
$$F_t - \mu = A(F_{t-1} - \mu) + \eta_t$$

对数似然函数为

$$\begin{aligned} \log L(y_1, y_2, \cdots . y_T) & \\ &= \log L(v_1, v_2, \cdots, v_T) \\ &= -0.5 \sum_{t=1}^{T} (-N \log(2\pi) - \log|V_t| - v_t' V_t^{-1} v_t) \end{aligned}$$

$F_{t|t-1}$ 和 V_t 的递推式由卡尔曼滤波算法得出

$$F_{t|t-1}=(I-A)\mu+AF_{t-1},\ \Sigma_{t|t-1}=A\Sigma_{t-1}A'+H$$
$$F_{t-1}=F_{t-1|t-2}+\Sigma_{t-1|t-2}\Gamma'(\lambda_1,\lambda_2)V_t^{-1}v_{t-1},$$
$$\Sigma_{t-1}=\Sigma_{t-1|t-2}-\Sigma_{t-1|t-2}\Gamma'(\lambda_1,\lambda_2)V_t^{-1}\Gamma(\lambda_1,\lambda_2)\Sigma_{t-1|t-2}$$

将对数似然函数关于参数极大化得出参数似然估计，将参数似然估计值带入似然函数的信息矩阵，计算出参数估计的协方差矩阵，并构造参数显著性检验的 t 统计值。

问题归类:

1）SAS/Base data 步数据整理;
2）用 SAS/ETS 中的过程 Proc SSM 实现状态空间模型建模。

教学过程:

step1：需要哪些预处理？——数据整理

i）外部数据转换为 SAS 数据集

从 Wind 获取的国债收益率月度数据使 Excel 表格形式的 csv（逗号分隔）数据

	year	6	12	18	24	30	36	42	48	
2	2006/1/6	1.462948	1.545676	1.955819	2.162519	2.194832	2.265424	2.381806	2.320397	2.36
3	2006/1/13	1.517894	1.549328	1.669094	1.988185	2.221033	2.340859	2.376589	2.225548	2.28
4	2006/1/20	1.44378	1.510635	1.723018	1.943192	2.204235	2.290294	2.420793	2.238592	2.3
5	2006/2/10	1.435784	1.688414	1.697246	2.151445	2.201575	2.270447	2.579783	2.227803	2.28
6	2006/2/17	1.628102	1.817423	1.680777	2.163503	2.260977	2.249519	2.678462	2.265328	2.31
7	2006/2/24	1.829735	1.77789	1.775476	2.231773	2.452024	2.237543	2.649412	2.338839	2.37
8	2006/3/3	1.752213	1.845058	1.806409	2.133991	2.241448	2.12369	2.597968	2.352183	2.38
9	2006/3/10	1.313178	1.487231	1.843615	2.162159	2.202237	2.146977	2.586295	2.342501	2.36
10	2006/3/17	1.627394	1.714023	1.975522	2.218211	2.187701	2.132596	2.660448	2.346213	2.36

首先用 proc import 过程将 Excel 表格转换为 SAS 数据集 yield22。程序代码为

```
proc import datafile="d:\mydata\yweek.csv"
out =yield22 dbms=csv replace;
run;
```

需要注意的知识点：由于原始数据为 csv 数据，需要在 proc import 语句中添加选项 dbms=csv。导入 SAS 后的 SAS 数据集的数据结构为

	year	_6	_12	_18	_24	_30	_36	_42	_4
1	2006-01-06	1.46294753	1.545676434	1.9558192	2.162519237	2.194832429	2.2654235	2.381806016	2.3203
2	2006-01-13	1.517894442	1.549327787	1.669094293	1.988184834	2.221032995	2.34085885	2.376589198	2.2255
3	2006-01-20	1.443779686	1.510635118	1.723018	1.943191704	2.204235341	2.290293581	2.420793009	2.2385
4	2006-02-10	1.435783602	1.688414215	1.69724586	2.151445183	2.201574693	2.270447015	2.57978324	2.2276
5	2006-02-17	1.6281019	1.817423293	1.680777269	2.163502995	2.260976889	2.249518857	2.678462377	2.2653
6	2006-02-24	1.829734545	1.777889515	1.775476292	2.231772981	2.452023801	2.237542577	2.649411944	2.338
7	2006-03-03	1.75221336	1.845058223	1.806409006	2.133990918	2.241448065	2.123690374	2.597968412	2.3521
8	2006-03-10	1.313178054	1.487231308	1.843614505	2.162158776	2.202237333	2.146977192	2.586295015	2.3425
9	2006-03-17	1.627393859	1.714023033	1.975522236	2.218210508	2.187701008	2.132596382	2.660447696	2.3462
10	2006-03-24	1.619900094	1.728668273	2.214861743	2.357222001	2.3055817	2.283865719	2.851829309	2.4111
11	2006-03-31	1.691377603	1.850381878	2.132324347	2.341794321	2.593702564	2.103200467	2.658934384	2.452

需要注意的知识点：输入数据 csv 表格的各列名称为数字（对应的期限），而 SAS 数据集变量名不允许使数字，因此，SAS 导入时会在原列名前添加下划线作为导入后 SAS 数据集变量名。

ii）重整重排（rearrange）

要采用 proc ssm 建立状态空间模型，需要将数据摆放为如下形式

	year	y	type	m
1	2006-01-06	1.46294753	1	6
2	2006-01-06	1.545676434	2	12
3	2006-01-06	1.9558192	3	18
4	2006-01-06	2.162519237	4	24
5	2006-01-06	2.194832429	5	30
6	2006-01-06	2.2654235	6	36
7	2006-01-06	2.381806016	7	42
8	2006-01-06	2.320396868	8	48
9	2006-01-06	2.362786051	9	54
10	2006-01-06	2.402141169	10	60
11	2006-01-06	2.499246885	11	66

即面板数据的形式，第一层识别变量为 year（交易日期），第二层识别变量为 type（债权类型：按期限不同划分），同时生成到期期限变量 m。

生成面板数据的程序代码为

```
data y22;
set yield22;
array term[*] _:;
do i=1 to dim(term);
y=term[i];
if i<=20 then m=i*6;
if i=21 then m=144;
if i=22 then m=168;
type=i;
output;
end;
keep year y type m;
run;
```

需要注意的知识点：（i）array 语句中用 term[*]定义一个不给下标的数组，数组长度由其后的变量个数确定。（ii）array 语句数组名 term[*]后面的变量名列表，采用缩略方式“**前缀：**”的形式给出。

step2：状态空间模型复杂性——模型的设定

状态空间模型的组成分为观测方程和状态方程，其设定较为复杂，往往需要编程才能完成。Data 步编程中的很多语句都可以直接在过程 proc ssm 中使用，例如 do-end 循环语句、数组等，极大方便了模型设定。

难点：如何设定状态方程常数项?

过程 Proc ssm 中的状态方程不允许由常数项，而大多数实际问题中的状态方程需要由常数项。例如，本案例中的三个状态变量分别代表了长期利率水平、利差和曲度，如果状态方差不带常数项，会导致三个变量的平均值等于 0，与实际情况不符。

解决方案：模型变换

具有显性常数项的状态方程为

$$\begin{bmatrix} L_t \\ S_{1,t} \\ S_{2,t} \\ C_t \end{bmatrix} = C + \Phi \begin{bmatrix} L_{t-1} \\ S_{1,(t-1)} \\ S_{2,(t-1)} \\ C_{t-1} \end{bmatrix} + \begin{bmatrix} \eta_{L,t} \\ \eta_{S_1,t} \\ \eta_{S_2,t} \\ \mu_{C,t} \end{bmatrix} ,\tau = m_1, m_2, \dots ,m_N$$

根据平稳性，对模型两边取期望得出

$$\begin{bmatrix}\mu_L\\ \mu_{S_1}\\ \mu_{S_2}\\ C_L\end{bmatrix}=C+\Phi\begin{bmatrix}\mu_L\\ \mu_{S_1}\\ \mu_{S_2}\\ C_L\end{bmatrix}\Rightarrow C=\begin{bmatrix}\mu_L\\ \mu_{S_1}\\ \mu_{S_2}\\ C_L\end{bmatrix}-\Phi\begin{bmatrix}\mu_L\\ \mu_{S_1}\\ \mu_{S_2}\\ C_L\end{bmatrix}$$

带入原模型得出

$$\begin{bmatrix}L_t-\mu_L\\ S_{1t}-\mu_{S1}\\ S_{2t}-\mu_{S2}\\ C_t-\mu_C\end{bmatrix}=\Phi\begin{bmatrix}L_{t-1}-\mu_L\\ S_{1,(t-1)}-\mu_{S1}\\ S_{2,(t-1)}-\mu_{S2}\\ C_{t-1}-\mu_C\end{bmatrix}+\begin{bmatrix}\eta_{L,t}\\ \eta_{S_1,t}\\ \eta_{S_2,t}\\ \mu_{C,t}\end{bmatrix},\tau=m_1,m_2,\dots,m_N$$

设新的状态变量向量为

$$\boldsymbol{\gamma}_t:=\begin{bmatrix}\gamma_{1t}\\ \gamma_{2t}\\ \gamma_{3t}\\ \gamma_{4t}\end{bmatrix}=\begin{bmatrix}L_t-\mu_L\\ S_{1t}-\mu_{S1}\\ S_{2t}-\mu_{S2}\\ C_t-\mu_C\end{bmatrix}\Rightarrow\begin{bmatrix}L_t\\ S_{1t}\\ S_{2t}\\ C_t\end{bmatrix}=\begin{bmatrix}\gamma_{1t}\\ \gamma_{2t}\\ \gamma_{3t}\\ \gamma_{4t}\end{bmatrix}+\begin{bmatrix}\mu_L\\ \mu_{S1}\\ \mu_{S2}\\ \mu_C\end{bmatrix}$$

新的状态方程为

$$\boldsymbol{\gamma}_t=\Phi\boldsymbol{\gamma}_{t-1}+\boldsymbol{\eta}_t$$

不再包含常数项，满足 proc smm 的要求。

同时需要将观测方程中的状态变量替换为变换后的状态变量$\boldsymbol{\gamma}_t$，得出新的观测方程为

$$y_t(\tau)=(\gamma_{1,t}+\mu_L)+\frac{1-e^{-\lambda_1\tau}}{\lambda_1\tau}(\gamma_{2,t}+\mu_{S1})+\frac{1-e^{-\lambda_2\tau}}{\lambda_2\tau}(\gamma_{2,t}+\mu_{S1})+\left[\frac{1-e^{-\lambda_1\tau}}{\lambda_1\tau}-e^{-\lambda_1\tau}\right](\gamma_{3,t}+\mu_C)+\epsilon_t(\tau)$$

为此，需要将常数向量$\boldsymbol{\mu}$设定为特殊的状态——误差项方差为 0 的随机游动!

$$\begin{bmatrix}\mu_{L,t}\\ \mu_{S1,t}\\ \mu_{S2,t}\\ \mu_{C,t}\end{bmatrix}=\begin{bmatrix}\mu_{L,t-1}\\ \mu_{S1,t-1}\\ \mu_{S2,t-1}\\ \mu_{C,t-1}\end{bmatrix}+\begin{bmatrix}e_{L,t}\\ e_{S1,t}\\ e_{S2,t}\\ e_{C,t}\end{bmatrix},\boldsymbol{e}_t\sim N(0,0)$$

据此得出状态方程的设定程序代码为

❋ 设定状态向量$\boldsymbol{\gamma}_t$的方程

```
state gamma(4) T(d)=(phi1 phi2 phi3 phi4) cov(g);
```

❋ 设定常数向量$\boldsymbol{\mu}$的方程

```
state mu(4) type=rw;/*type=rw不带方差选项意味着方差为0*/
```

step3：观测方程设定——模型的设定

本案例中观测方程的设定仍然较为复杂，主要是因为 22 个不同到期日的债权收益率需要设定不同的误差项方差。

难点：如何设定观测方程误差项方差?

为了描述 22 个不同债权品种的波动特性，需要对每个债权模型误差项设定方差参数，逐个设定不仅需要较多的重复编程语句，也很难实现和不同债权对应。

解决方案：采用数组和 do-end 循环

首先定义 22 维数组，再将 22 个方差参数作为数组元素，然后采用 do-end 语句实现方差参数和债权品种的匹配。程序代码为

❋ 设定观测方程误差项方差

```
parms sigma1-sigma22/lower=1.e-4;
array s_array(22) sigma1-sigma22;
```

```
do i=1 to 22;
if (type=i) then sigma = s_array[i];
end;
❋ 设定观测方差误差项（22维）
irregular wn variance=sigma;
```

由此得出观测方程设定的程序代码

```
❋ 设定因子载荷
Z1= 1.0;
tmp1 = exp(-lmd1*m);
tmp2 = exp(-lmd2*m);
Z2 = (1-tmp1)/(lmd1*m);
Z3 = (1-tmp2)/(lmd2*m);
Z4 = ( 1-tmp1-lmd1*m*tmp1)/(lmd1*m);
❋ 设定状态变量和常数项
state gamma(4) T(d)=(phi1 phi2 phi3 phi4) cov(g);
state mu(4) type=rw;
❋ 设定观测方程成分
comp gammaComp =(Z1-Z4)*gamma;
comp muComp =(Z1-Z4)*mu;
❋ 设定观测方程
model y= muComp gammaComp wn;
```

step4：模型的设定

在状态方程和观测方程设定的基础上，完成整个模型设定，包括参数的命名和界约束（bound restriction）设定、输入输出设定等。完整程序代码为：

```
proc ssm data=book.yield22 opt(tech=dbldog maxiter=300);
lmd1=0.2;lmd2=2.79;
parms phi1-phi4/lower=-0.99 upper=0.99;
parms sigma1-sigma22/lower=1.e-4;
/* Part I: set 22 error term in measurement eq.*/
array s_array(22) sigma1-sigma22;
do i=1 to 22;
if (type=i) then sigma = s_array[i];
end;
irregular wn variance=sigma;
/* Part II: set factor loadings*/
Z1= 1.0;
tmp1 = exp(-lmd1*m);
tmp2 = exp(-lmd2*m);
Z2 = (1-tmp1)/(lmd1*m);
Z3 = (1-tmp2)/(lmd2*m);
Z4 = ( 1-tmp1-lmd1*m*tmp1)/(lmd1*m);
/* Part III: set state*/
```

```
state gamma(4) T(d)=(phi1 phi2 phi3 phi4) cov(g);
state mu(4) type=rw;
/* Part IV: set measurement eq.*/
comp gammaComp =(Z1-Z4)*gamma;
comp muComp =(Z1-Z4)*mu;
model y= muComp gammaComp wn;
/* Part V: set components for output*/
comp gamma1 = gamma[1];
comp gamma2 = gamma[2];
comp gamma3 = gamma[3];
comp gamma4 = gamma[4];
output out=dnsFor pdv;
run;
```

step5：模型是否合适？——效果评价

模型是否合适，主要是看模型是否充分捕捉了数据中的相关性。常用的衡量标准时模型检测（model checking）：检测观测方程误差项的估计——观测方程的估计残差是否白噪声！

输出数据集 dnsFor 中的变量 Residual_y 为观测方程估计残差。

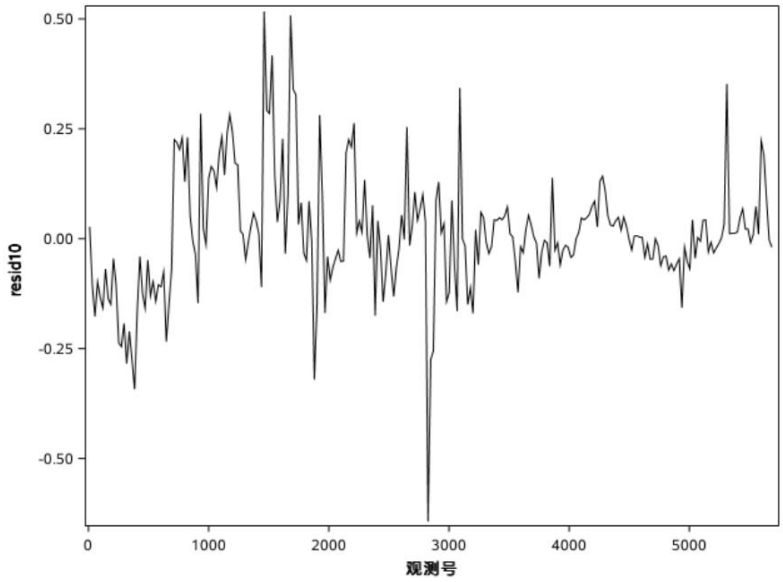
VIEWTABLE: Work.Dnsfor

	Z4	Obs	y	FORECAST_y	RESIDUAL_y	St
1	0.2811439448	1	1.46294753	.	.	
2	0.2881495662	2	1.545676434	.	.	
3	0.2428641324	3	1.9558192	.	.	
4	0.1983890556	4	2.162519237	.	.	
5	0.1637747891	5	2.194832429	2.2665007299	-0.071668301	0.51
6	0.1380386106	6	2.2654235	2.2240179949	0.0414055051	0.36
7	0.1187959818	7	2.381806016	2.2794634134	0.1023426026	0.30
8	0.1040918829	8	2.320396868	2.3935522621	-0.073155394	0.2
9	0.0925703042	9	2.362786051	2.3432583285	0.0195277225	0.17
10	0.0833266771	10	2.402141169	2.3752045525	0.0269366165	0.15
11	0.075755585	11	2.499246885	2.4074815542	0.0917653308	0.14

用 data 步生成 22 个债券收益率的估计残差序列，代码如下

```
%macro resid;
%do i=1 %to 22;
data resid&i;
set dnsfor;
if type=&i then resid&i=residual_y;
keep obs type m y resid&i;
if resid&i;
run;
proc sort data=resid&i;
by obs;
run;
%end;
%mend;
%resid;
```

可以用 proc sgplot 画出残差时序图，以 m=60（5 年）期限债券的残差为例



为更为严格判断是否为白噪声，可采用 proc arima 进行白噪声检验，程序代码为

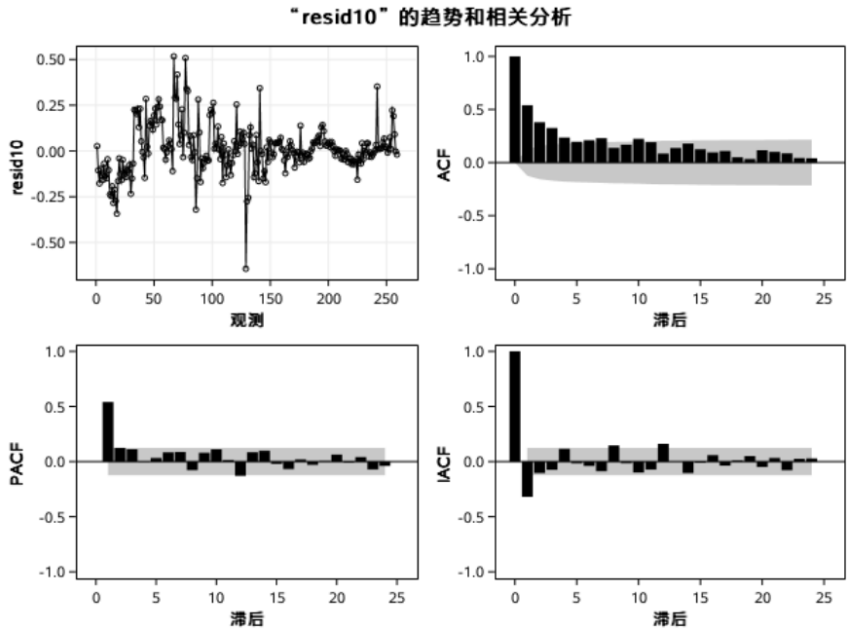
```
proc arima data=resid10;
identify var=resid10;
run;
```

运行结果显示

白噪声的自相关检查

至滞后	卡方	自由度	Pr > 卡方	自相关					
6	179.41	6	<.0001	0.541	0.380	0.326	0.236	0.196	0.211
12	232.03	12	<.0001	0.229	0.138	0.169	0.224	0.192	0.085
18	256.99	18	<.0001	0.137	0.179	0.125	0.096	0.109	0.050
24	267.20	24	<.0001	0.034	0.117	0.101	0.087	0.043	0.040

拒绝白噪声原假设。自相关函数 ACF 和偏自相关函数 PACF 图为



显示出存在自相关。

从残差序列图看出，序列自相关的最可能原因是误差项可能存在条件异方差 GARCH。要消除序列相关，需要引入带 GARCH 结构的状态空间模型，而现有的软件没有这种功能。

总结讨论:

因子模型：将高度相关的高维时间序列表示为较少个公因子（common factors）和个体成分（idiosyncratic componet）的和，能够显著降低模型维度。本案例中的动态因子利率期限结构模型，将 22 维国债收益率向量表示为 3 个因子（水平、斜率和曲度）的状态空间模型，解决了收益率曲线构造中的维数灾难（curse of dimension）问题，是一种金融建模中常用的手段和方法。

软件应用：SAS/ETS 软件中的过程 proc ssm 具有强大功能，能够设定和估计多种状态空间模型。本案例用 proc ssm 过程建立中国国债收益率曲线。应用的重点在于结合 data 步编程语言设定状态方程和观测方程, 难点在于如何将实际中的模型变形为可用软件估计的模型——状态方程常数项的处理。

思考题:

如何用 proc ssm 判断数据中是否有异常值（outlier）？异常值会否影响模型的拟合效果？如果发现异常值，你能够判断产生异常值的原因？如何处理？

案例 3：资产定价——时变系数 CAPM 模型

案例背景：

资本资产定价模型 CAPM（Capital Assets Pricing Model）是金融学中权益资产的顶级模型。CAPM 模型将金融资产收益率表示为市场收益率和个体冲击的线性函数，给出了风险和预期收益的关系。CAPM 把金融资产的风险分解为系统风险（systematic risk）和个别风险（idiosyncratic risk），据此认为只有系统性风险才能获得风险补偿（risk premium），非系统性风险可通过分散投资进行消除，不能获取风险补偿。CAPM 模型在投资组合管理中具有重要应用，在股票市场中，CAPM 模型中的贝塔系数成为股票的重要特征。

案例描述：—— 提出问题

传统模型中，常把 CAPM 模型样本期内的贝塔系数视作常数，采用时间序列回归的方法进行估计，并据此进行投资组合管理。由于市场条件和上市公司情况的变化，上市公司股票的特征也会发生变化，采用常贝塔系数和阿尔法系数进行投资组合管理，不能及时根据市场变化和公司层面的变化进行及时有效的调整，很难得实时的有效投资组合，降低了投资效率。

把 CAPM 模型的回归系数时变化，采用高斯线性状态空间形式的 CAPM 模型，通过 Kalman 滤波和平滑获得时变的贝塔系数序列，是一种较为复合实际的方法。但通常情况下，状态空间模型的观测方程误差项常常设定为同方差，而自回归条件异方差 ARCH 或者 GARCH 是股票收益时间序列的三大特征之一，将观测方差误差项设定为同方差，不能有效捕捉金融时间序列数据中的波动群聚和厚尾性特征。

本案例采用 SAS/ETS 中的 proc ssm 过程建立观测方程误差项具有 GARCH 结构的线性高斯状态空间模型，作为 CAPM 模型对招商银行在 2016 年 1 月 5 日到 2020 年 12 月 31 日样本期内的时变贝塔系数进行估计。模型估计和推断采用上证指数（000001）和招商银行（600036）样本期内日收益率数据，样本量为 1159 个样本。

要估计的模型具有状态空间模型的形式

$$r\_z_t = \alpha_t + \beta_t r\_sh_t + \epsilon_t, \epsilon | \mathcal{F}_{t-1} \sim N(0, \sigma_t^2)$$

$$\begin{bmatrix} \alpha_t \\ \beta_t \end{bmatrix} = \begin{bmatrix} c_1 \\ c_2 \end{bmatrix} + \Phi \begin{bmatrix} \alpha_{t-1} \\ \beta_{t-1} \end{bmatrix} + \begin{bmatrix} \eta_t \\ \eta_t \end{bmatrix}$$

如果$\begin{bmatrix} \alpha_t \\ \beta_t \end{bmatrix} = \begin{bmatrix} \alpha \\ \beta \end{bmatrix}$，则模型为常系数CAPM模型，可看作时间序列回归模型通过proc autoreg进行估计。如果$\sigma_t^2 = \sigma^2$，则表明观测方差误差项不存在ARCH效应。

问题归类：

1）时变系数回归模型的状态空间模型估计；
2）在过程 Proc SSM 实现观测方程误差项的 GARCH 建模和估计。

教学过程：

step1：常系数贝塔合理吗？——滚动窗口回归观察

为考察招生银行贝塔系数的时变性特征，用 proc autoreg 过程对样本期内样本进行滚动窗口回归，窗口长度为 200 个交易日，窗口之间距离为 20 个交易日。具体做法为：

1. 用第 1 个样本到 200 个样本进行回归，得出第一个贝塔系数$\hat{\beta}^{(1)}$;
2. 用第 21 个样本到 222 个样本进行回归，得出第二个贝塔系数$\hat{\beta}^{(2)}$;

……

k. 用第$k*20+1$个样本到第$200+k*20+1$个样本进行回归，得出第 k 个贝塔系数$\hat{\beta}^{(k)}$

尝试避免采用宏，这里采用 data 步中调用例程 execute()的方法实现滚动窗口回归。首先生成用于 proc autoreg 的各个子样本数据集的起止观测序列号。data 步程序为

```
data subset;
do i=1 to 100 while(endobs<=1159);
temp1=(i-1)*20;
temp2=temp2+200;
startobs=put(temp1,4.);
endobs=put(temp2,4.);
i1=put(i,3.);
output;
end;
run;
```

以数据集 subset 为依托，在 data 步内置循环中调用过程 proc autoreg，并将参数估计结果输出为数据集 capm_i。单次的 proc autoreg 程序为

```
proc autoreg data=sh_bys_zsyh_r outest=c;
model r_z=r_sh;
run;
```

其中数据集 sh_bys_zsyh_r 为 work 逻辑库中数据集，其中的变量 r_sh 和 r_z 分别为上证指数日收益率和招商银行日收益率。proc autoreg 语句中选项 outest=c 将参数估计结果输出为数据集 c，结构如下

	NAME	_MSE_	_SSE_	Intercept	r_z	r_sh
1		0.0001790392	0.2071484055	0.0007584593	-1	0.8277968047

其中 Intercept 变量值为回归常数项——CAPM 模型阿尔法系数，r_sh 变量值为斜率系数——CAPM 模型中贝塔系数。

将上述程序段嵌入数据集 subset 的 data 步内置循环中，通过调用例程 execute( )实现对每个子样本的 proc autoreg 操作，子样本的确定通过 where 数据集选项或者 where 语句来完成。为用 where 语句选定滚动窗口操作中的不同子样本区间，需要首先在数据集 sh_bys_zsyh_r 中生成表示观测序号的变量 obs，程序代码为

```
data sh_bys_zsyh_r;
set sh_bys_zsyh_r;
obs=put(_n_,4.);
run;
```

最后实现 proc autoreg 对滚动样本窗口的操作，程序代码如下：

```
data _null_;
set subset;
txt1=cats('proc autoreg data=sh_bys_zsyh_r
outest=c',i1,';');
```

```
txt2='where '||startobs||'<=obs<='||endobs||';';
txt3='model r_z=r_sh;';
txt4='run;';
txt=txt1||txt2||txt3||txt4;
call execute(txt);
run;
```

运行结果提示

```
ERROR: WHERE 子句运算符要求兼容的变量。
NOTE: 由于出错，SAS 系统停止处理该步。
```

难点：问题出在哪里？

根据错误提示，再次查看例程 execute( )的帮助发现，execute( )自变量中的字符串表达式，只能是单引号括住的字符常量、data 步数据集中的字符型变量，并且字符变量的取值必须是文本而不能是数字！程序中的第二个字符串表达式

```
txt2='where '||startobs||'<=obs<='||endobs||';';
```

中的变量 starobs 和 endobs 尽管是字符变量，但其取值是数字。

解决方案：采用宏

用宏循环%do-%end，在 where 语句中用循环指标宏变量 i 选定参与 proc autoreg 操作的观测。为此，需要在输入数据集中生成数值变量 obs，程序代码为

```
data sh_bys_zsyh_r;
set sh_bys_zsyh_r;
obs_n_;
run;
```

采用宏循环实现滚动窗口的 proc autoreg 处理，代码为

```
%macro autoreg;
%do i=1 %to 49;
proc autoreg data=sh_bys_zsyh_r outest=c&i noprint;
where ((&i-1)*20+1)<=obs<=((&i-1)*20+20);
model r_z=r_sh;
run;
%end;
%mend;
%autoreg;
```

为避免输出过多内容影响运行速度，在 proc autoreg 语句中添加选项 noprint。程序运行后生成 49 个参数估计结果数据集 c1-c49。用 data 步 set 语句将这些数据集上下叠放合并，只保留变量 intercept 和 r_sh，将其改名为 alpha 和 beta 形成数据集 c，代码为

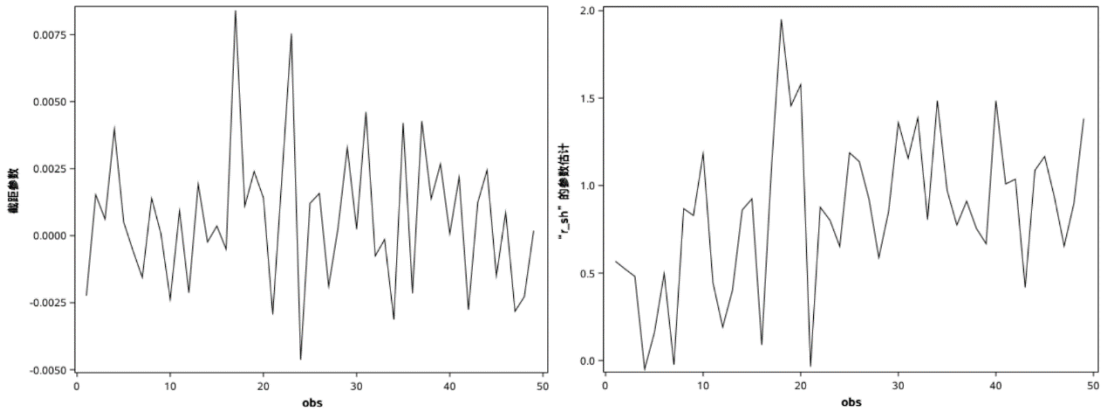
```
data c(rename=(intercept=alpha r_sh=beta));
set c1-c49;
keep intercept r_sh;
run;
```

用 proc sgplot 画出 alpha 和 beta 的时序图，程序为

```
data c;
set c;
```

```
obs=_n_;
run;
proc sgplot data=c;
series x=obs y=alpha;
run;
proc sgplot data=c;
series x=obs y=beta;
run;
```

输出结果为



可以看出，无论是 alpha（截距参数）还是 beta（r_sh 的参数估计）都显出明显的时变特征，况且其时变特征具有一阶自回归的特点。

step2：误差项同方差合理吗？——残差 ARCH 检验

由于异方差不影响模型参数极大似然估计的一致性，可先用 proc ssm 建立同方差状态状态空间模型，然后对观测方程残差进行 ARCH 检验。

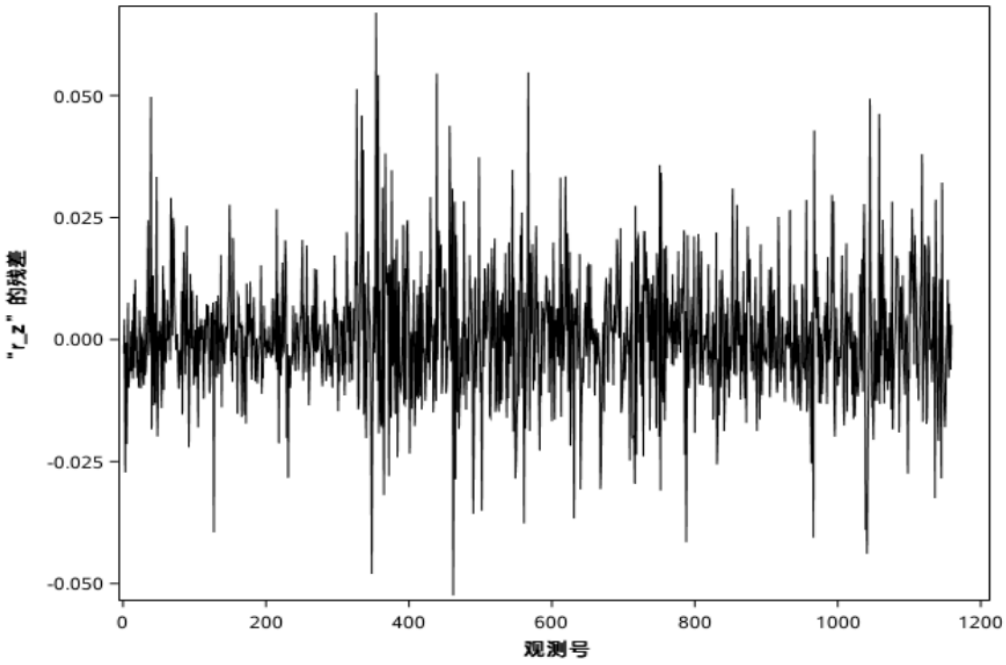
首先建立同方差模型，程序代码为

```
proc ssm data=book.sh_bys_zsyh_r;
irregular e;
state s(2) cov(g) type=varma(p(D)=1);
v1=1;
component s1=(v1 r_sh)*s;
model r_z=s1 e;
output out=jg;
run;
```

定义观测方差误差项 e 的语句 irregular e；没有带任何选项，表明白噪声 e 的方差为常数。输出数据集 jg 中变量 residual_r_z 为观测方差估计残差。用 proc sgplot 画出的时序图的程序为

```
proc sgplot data=jg;
series x=obs y=residual_r_z;
run;
```

画出的时序图为



从图中可以看出明显的 ARCH 效应。

为更准确判断观测方差误差项是否有 ARCH 效应，用 proc autoreg 对序列 residual_r_z 进行 ARCH 检验，程序为

```
proc autoreg data=jg;
model residual_r_z=/ARCHTEST;
run;
```

结果显示为

基于 OLS 残差的 ARCH 扰动检验

阶数	Q	Pr > Q	LM	Pr > LM
1	12.9165	0.0003	12.9300	0.0003
2	16.3147	0.0003	15.1251	0.0005
3	28.2048	<.0001	24.7686	<.0001
4	39.6917	<.0001	31.9385	<.0001
5	45.7339	<.0001	34.5029	<.0001
6	60.8485	<.0001	43.5545	<.0001
7	61.5700	<.0001	43.7459	<.0001
8	62.8157	<.0001	43.8158	<.0001
9	80.6677	<.0001	54.2856	<.0001
10	80.8212	<.0001	55.2540	<.0001
11	82.0914	<.0001	55.5089	<.0001
12	82.1144	<.0001	56.4758	<.0001

检验结果表明，采用 Engle 的 LM（拉格朗日乘子）法检验拒绝无 ARCH 效应的原假设，表明存在明显的 ARCH 效应。

step3：如何设定 GARCH 误差项？——proce ssm 中的编程语句

在 proc ssm 中将观测方程误差项的方差设定为 GARCH 模型的困难在没有对应的选项。

难点：proc ssm 没有 GARCH 选项

proc ssm 设定的状态空间模型中，观测方差误差项和状态方程误差项都没有 GARCH 选项，不能直接设定误差项方差的 GARCH 模型。

解决方案：采用编程语句

尽管没有 GARH 选项，irregular 语句的选项 variance=variable 可以将误差项方差设定为变量，而利用 proc ssm 中的编程语句（programming statements）可以将误差项方差设定为 GARCH 模型。GARCH 模型的一般形式为

$$\sigma_t^2 = \omega + \theta_1\sigma_{t-1}^2 + \theta_2 y_{t-1}^2, 0 < \theta_1, \theta_2 < 1, \theta_1 + \theta_2 < 1$$

利用 proc ssm 中的编程语句设定 GARCH 模型，程序代码为

```
proc ssm data=book.sh_bys_zsyh_r;
parms theta1 theta2/lower=(1e-3) upper=(0.999);
parms omega/lower=(1e-3);
retain sigma 0;
sigma=omega+theta1*sigma+theta2*((lag(r_z))**2);
irregular e variance=sigma;
state s(2) cov(g) type=varma(p(D)=1);
v1=1;
component s1=(v1 r_sh)*s;
model r_z=s1 e;
output out=jg;
run;
```

运行后，信息窗口出现错误提示：

```
ERROR: Missing or negative variance value provided in e.  No further calculations will be done.
```

提示在 irregular 语句中定义的方差取值为 0 或者缺失值。查看程序发现：1）retain 语句将 omega 初值设定为 0；2）在语句

```
sigma=omega+theta1*sigma+theta2*((lag(r_z))**2);
```

中滞后运算 lag(r_z)导致第一个观测值的滞后为缺失值。

错误：初始值和滞后缺失

为此做两个方面改进：

修改：omega 单独计算，添加滞后变量

观测方差误差项的无条件方差可以用提前的同方差状态空间模型观测方差估计残差计算出，根据 GARCH 模型得出

$$\sigma_\epsilon^2 := Var(\epsilon_t) = \frac{\omega}{1-\theta_1-\theta_2} \Rightarrow \omega = \sigma_\epsilon^2(1-\theta_1-\theta_2)$$

由此得出

$$\hat{\omega} = \hat{\sigma}_{\epsilon}^{2}(1-\theta_1-\theta_2)$$

这样就可以将ω表示为其他参数的函数，这种方法称为 variance-tracking。对输出数据集 jg 中变量 residual_r_z 求方差得出

```
proc means data=jg var;
var residual_r_z;
run;
```

结果为

分析变量: RESIDUAL_r_z “r_z”的残差
方差
0.000177168

此外，在数据集 sh_bys_zsyh_r 中生成 r_z 的一阶滞后变量，且让第一个观测值取 0

```
data sh_bys_zsyh_r;
set book.sh_bys_zsyh_r;
r_z_1=lag(r_z);
if _n_=1 then r_z_1=0;
run;
```

以 sh_bys_zsyh_r 为输入数据集，以

$$\hat{\omega} = 0.000177168 * (1-\theta_1-\theta_2)$$

编写 proc ssm 程序，即

```
proc ssm data=sh_bys_zsyh_r;
parms theta1 theta2/lower=(1e-3) upper=(0.999);
omega=(1-theta1-theta2)*0.000177168;
retain sigma 0.000177168;
sigma=omega+theta1*sigma+theta2*(r_z_1**2);
irregular e variance=sigma;
state s(2) cov(g) type=varma(p(D)=1);
v1=1;
component s1=(v1 r_sh)*s;
model r_z=s1 e;
output out=jg;
run;
```

运行后，信息窗口出现错误提示：

```
ERROR: Missing or negative variance value provided in e.  No further calculations will be done.
```

提示在 irregular 语句中定义的方差取值为 0 或者缺失值。查看程序发现：需要将θ_1,θ_2的取值约束为$0 < \theta_1 + \theta_2 < 1$才能保证方差 sigma 的非负性。

错误：方差值为负

困难：无线性约束选项和语句。

proc ssm 对模型参数只有上下界约束，通过在 parms 中斜杠后选项 lower=()和 upper=

（ ）实现，没有参数间的线性约束或者非线性约束选项。

出路：两步法

两步法将具有 GARCH 结构观测误差项的状态空间模型分两步估计：第一步对无 GARCH 结构的观测项误差的模型进行估计，然后对残差项建立 GARCH 模型，据此计算出误差项条件方差σ_t^2；第二步将σ_t^2作为已知变量引入 proc ssm 过程中，估计具有 GARCH 观测误差项的模型。

第一步：对无 GARCH 结构观测误差的状态空间模型运行 proc ssm 后，输出数据集 jg 中变量 residula_z_r 建立 GARCH(1,1)模型。采用 proc autoreg 模型，程序代码为

```
proc autoreg data=jg;
model residual_r_z=/GARCH=(p=1,q=1);
output out=var ht=sigma;
run;
```

参数估计结果为

参数估计

变量	自由度	估计	标准误差	t 值	近似 Pr > |t|
Intercept	1	0.000524	0.000367	1.43	0.1531
ARCH0	1	0.0000100	2.1983E-6	4.56	<.0001
ARCH1	1	0.0752	0.0140	5.38	<.0001
GARCH1	1	0.8687	0.0220	39.51	<.0001

得出 GARCH 模型为

$$\sigma_t^2 = 1e^{-4} + 0.075\epsilon_{t-1}^2 + 0.8687\sigma_{t-1}^2$$

输出数据集 var 中变量 hd 为按此公式计算的条件方差序列，一同输出的还有变量 r_z 等。

VIEWTABLE: Work.Var

	sigma	Obs	r_z	FORECAST_r_z	RESIDUAL_r_z	StdErr_r_z	Lower
1	0.0001777415	1	-0.002897713	0	-0.002897713	0.0131137393	-0.0257
2	0.0001653072	2	0.0046323186	0.0004518818	0.0041804368	0.0218790504	-0.0424
3	0.0001546304	3	-0.037673672	-0.010436406	-0.027237266	0.0361781596	-0.0813
4	0.0002022707	4	0.0083632507	0.009008846	-0.000645595	0.0136914227	-0.0178
5	0.0001858377	5	-0.046888345	-0.025453586	-0.021434759	0.0153384845	-0.0555
6	0.0002076981	6	-0.003747662	0.0009122159	-0.004659878	0.0130206482	-0.0246
7	0.0001924692	7	-0.006907406	-0.01446869	0.0075612842	0.0131957877	-0.0403
8	0.000180942	8	0.001259446	0.0111087235	-0.009849278	0.0135990442	-0.0155
9	0.0001752945	9	-0.026141112	-0.019336995	-0.006804117	0.0137096746	-0.0462
10	0.0001663375	10	-0.0038835	0.0022474651	-0.006130965	0.0130662389	-0.0233
11	0.0001578495	11	0.0186320841	0.0169289471	0.001704037	0.0142663987	-0.011

第二步：将 work.var 和 sh_bys_zsyh_r 进行合并（merge），保留变量 sigma、obs、r_z 和 r_sh，生成数据集 sh_zsyh_r:

```
data sh_zsyh_r;
merge sh_bys_zsyh_r var;
keep tdate r_sh r_z sigma;
run;
```

以 sh_zsyh_r 为输入数据集，采用 proc ssm 估计变系数 CAPM 模型，在 irregular 语句中用选项 variance=sigma 刻画观测方程误差项的 GARCH 结构。程序代码为

```
proc ssm data=sh_zsyh_r;
irregular e variance=sigma;
state s(2) cov(g) type=varma(p(D)=1);
v1=1;
component s1=(v1 r_sh)*s;
model r_z=s1 e;
output out=jg;
run;
```

总结讨论:

时变系数：市场条件和上市公司情况随时间发生变化，导致常贝塔系数的 CAPM 模型不再是合适的资产定价模型。把 CAPM 模型回归系数时变化，得出线性高斯状态空间模型形式的 CAPM 模型。此外，考虑到风险资产收益分布的厚尾性和波动群聚性，需要在字长方程误差项中引入 GARCH 结构。

灵活应用：SAS/ETS 软件中的过程 proc ssm 虽然具有强大功能，也很难满足所有实际需要。当不能直接采用给定的功能进行数据分析时，要采用迂回的方法，采用多种方法结合解决实际问题。本案例中在观测方程误差项中添加 GARCH 结构的方法，会带给我们很多启发。灵活运用的前提时对模型的透彻理解和对软件功能的全面把握，本案例的解决方法，需要对状态空间模型、GARCH 模型十分熟悉，同时又要求对 proc ssm、proc autore 以及 SAS 中 data 步的功能又全面的掌握。

思考题:

本案例中采用的两步法会影响参数估计的一致性吗？为什么？会影响参数估计的效率吗？为什么？还有更好的解决方法吗？

案例 4：网页表格数据爬取[①]

案例背景：

当今的互联网时代，网络是人们发布和获取信息的主要方法，从网页上获取数据已成为大数据分析的基本技术和技巧。流行软件如 Python、R 等都有专门用于网页信息爬取（Web crawling）的程序或者函数。SAS 也发布了专门用于网页爬取和文本分析的产品。但专门用于网页表格数据爬取的程序并不多见。本案例采用 data 步获取网页信息并对其中隐含的表格数据进行爬取，生成 SAS 数据集。

案例描述：—— 提出问题

很多网页发布的数据以网页表格形式呈现，并不提供 Excel 等形式的文表格文件下载。例如中国外汇交易中心网页 http://www.shibor.org/shibor/web/html/shibor.html 提供的上海银行间同业拆放利率 shibor 呈现为

2021-10-14 11:00

期限	Shibor(%)	涨跌(BP)
O/N	2.1060	5.10
1W	2.1980	0.40
2W	2.3000	10.10
1M	2.3590	0.20
3M	2.4230	0.30
6M	2.4980	0.30
9M	2.6530	0.20
1Y	2.7250	0.80

并没有提供表格数据下载接口。本案例采用 SAS 软件的 data 步编程对网页表格数据进行爬取，生成 SAS 数据集文件。

问题归类：

1）data 步读取网页文件；
2）data 读取外部文件并进行数据表格爬取。

教学过程：

step1：如何读取网页数据？——infile 语句选项 url

为了爬取网页表格数据，需要把网页数据保存为文本文件。通过 data 的 infile 语句及其选项 url，可以将网页作为外部文件用 input 语句进行读取。以下程序读取网页 http://www.shibor.org/shibor/web/html/shibor.html 内容，生成 txt 文件 d:\mydata\myfile.txt.

程序 1：

```
filename cm url
'http://www.shibor.org/shibor/web/html/shibor.html';
```

[①] 本案例参考 Rick Langston 的报告“Creating SAS Data Sets from HTML Table Definition”, SAS Global Forum 2009, Paper 052-2009. SAS Institute Inc.

```
filename myfile 'd:\mydata\myfile.txt';
data _null_;
infile cm recfm=f lrecl=1 end=eof;
file myfile recfm=f lrecl=1;
input @1 x $char1.;
put @1 x $char1.;
if eof;
call symputx('filesize',_n_);
run;
```

程序中的关键点：

1. 为了准确读取网页的全部内容，infile 语句采用了选项 recfm=f 和 lrecl=1，第一个选项要求在读取外部数据时按固定长度确定数据行，此时外部数据的行结束符不在起作用，只要读够给定的数据长度，则认为当前数据行结束，lrecl=1 则规定数据行的长度为 1，即每次读取 1 个字符。如果遇到汉字等 utf-8 编码的字符，则逐个读取。input 语句中指针控制符@1 要求每次读取都从当前数据行的第一个字符读取，变量 x 后的格式$char1.要求读取 1 个字符给变量 x，空格也作为字符读取。

2. 为了计算整个文件的字符个数供下一个 data 使用，用例程调用 call symputx( )将 data 步内置循环次数_n_的值赋给宏变量 filesize。

3. 输出为外部文本 myfile 时，采用 file 选项 recfm=f 和 lrecl=1 和 put 语句指针可控制@1 及变量输出格式$char1.，保证了生成的外部文件和网页文件文本排列相同。

生成的外部文件为（为节省空间，只列出片段）

```
<table width="474" border="0" cellspacing="0" cellpadding="0"  >
<tr>
<td><table  width="100%"  height="20"  border="0"  cellpadding="0"
cellspacing="0">
 <tr>
 <td   align="right"   valign="top"   class="infoTitleW">2021-10-14
11:00  </td>
</tr>
</table></td>
</tr>
 <tr>
<td  align="center"><table  width="95%"  border="0"  cellspacing="0"
cellpadding="0">
 <tr>
 <td width="10%" align="center" class="infoTitleU"> </td>
  <td width="15%" align="center" class="infoTitleU">期限</td>
  <td                    width="30%"                    align="center"
class="infoTitleU">    Shibor(%)</td>
 <td width="15%" align="right" class="infoTitleU"> </td>
   <td width="30%" align="left" class="infoTitleU">涨跌(BP)</td>
 </tr>
 <tr>
  <td height="5" colspan="5" bgcolor="c2d7e9"></td>
```

```
   </tr>
  </table>
  <table  width="95%"  border="0"  cellspacing="0"  cellpadding="0"
class="shiborquxian" >
       <tr>
 <td       width="10%"       align="center"       height="30"><img
src="/shibor/web/html/images/newimages/shibor_arrow.gif"
width="11" height="12"></td>
 <td         width="15%"         align="center"><a         href="#"
onClick="window.open('/shibor/shiborChartShow.do?termId=O/N','走势
图     ','width=800,height=550,top=120,left='+(screen.availWidth/2-
405)+',toolbar=no,menubar=no,scrollbars=yes,
resizable=yes,location=no,   status=no')"   title="Shibor  走  势  图
"   ><font color=blue>O/N</font></a></td>
 <td width="30%" align="center">2.1060</td>
 <td                 width="15%"                 align="right"><img
src="/shibor/web/html/images/newimages/downicon.gif"    width="11"
height="11" />    </td>
 <td width="30%" align="left">  5.10</td>
       </tr>
       <tr>
 <td       width="10%"       align="center"       height="30"><img
src="/shibor/web/html/images/newimages/shibor_arrow.gif"
width="11" height="12"></td>
 <td         width="15%"         align="center"><a         href="#"
onClick="window.open('/shibor/shiborChartShow.do?termId=1W',' 走 势
图     ','width=800,height=550,top=120,left='+(screen.availWidth/2-
405)+',toolbar=no,menubar=no,scrollbars=yes,
resizable=yes,location=no,   status=no')"   title="Shibor  走  势  图
"   ><font color=blue>1W</font></a></td>
 <td width="30%" align="center">2.1980</td>
 <td                 width="15%"                 align="right"><img
src="/shibor/web/html/images/newimages/downicon.gif"    width="11"
height="11" />    </td>
 <td width="30%" align="left">  0.40</td>
       </tr>
       <tr>
 <td       width="10%"       align="center"       height="30"><img
src="/shibor/web/html/images/newimages/shibor_arrow.gif"
width="11" height="12"></td>
 <td
```

step2：寻找网页表格标签

HTML 文件常以表格 table 方式组织数据，其中用<table、/table 标识表格的开始和结

束，<tr 和/tr 标识表格行的开始和结束、<td 和/td 标识单元格的开始和结束。要爬取表格数据，需要首先识别表、表中的行、行中的单元格的开始和结束，并对出现的次数进行统计，

data 步将文本文件 myfile.txt 的内容作为一行，用 input 语句多次逐字符扫面，寻找特定字符所在的列，同时计算字符出现的次数。设变量 tag 的取值是要寻找的字符（例如 tag='<table'、tag='</table'等），则如下程序段将 tag 的内容和在文件 myfile.txt 的列数输出为变量 text 和 col，并生产变量 obscount。

程序段 1：

```
obscount=0;
input @1 @;
do while(1);
  input @(trim(tag)) @;
  if c>&filesize then leave;
  col=c;
  obscount+1;
  output;
end;
```

其中do-while循环一致进行，直到读完所有列（c>&filesize）才跳出循环（leave）。循环中的input语句采用选项@char将读取指针指向char之后的第一个字符，选项@使得do-while循环内的下一次input从上次input语句读取后的指针开始读取。由于input没有读取给变量，读取完后的指针位置就是tag字符后的第一个字符。

do-while循环之前的input语句，是搜寻完一个tag后，开始搜寻下一个tag时数据指针的初始化：即仍然从第一个字符开始读取！

将 tag 分别为'<table'、'<\table'、'<TABLE'、'<\TABLE'、'<tr'、'<\tr'、'<TR'、'<\TR'和'<td'、'<\td'、'<\TD'、'<\TD'的搜算程序段嵌入 do-end 循环中，得出表格标签搜索程序

程序 2：

```
data tabletag(keep=text col);
infile myfile recfm=f lrecl=&filesize. column=c missover;
length tag $8 ;
closetags='N';
failure=0;
array whichtag{3} $8 _temporary_;
do i='table','tr','td';
 tag='<'||i;link readfile; n_lower_open=obsct;
 tag=upcase(tag);link readfile; n_upper_open=obsct;
 tag='</'||i;link readfile; n_lower_close=obsct;
 tag=upcase(tag);link readfile;n_upper_close=obsct;
 nobs+n_lower_open+n_upper_open+n_lower_close+n_upper_close;
 if i^='table' and n_lower_close+n_upper_close>0 then
    closetags='Y';
 if i='table ' then
   do;
   ntables=n_lower_open+n_upper_open;
   if ntables=0 then
     do;
```

```
      put 'Error: There are no tables defined in the HTML.';
      failure=1;
      end;
     else if n_upper_close+n_lower_close=0 then
      do;
      put 'Error: There are no closing tags for tables in the
     HTML.';
      failure=1;
      end;
     end;
      if n_upper_open>0 then
      do;
      if i='table' then call symput('tabletag',trim(upcase(i)));
      if i='tr' then call symput('trtag',trim(upcase(i)));
      if i='td' then call symput('tdtag',trim(upcase(i)));
     end;
  end;
  call symput('closetag',closetags);
  call symput('nobs',cats(nobs));
  call symput('ntables',cats(ntables));
  if failure then abort;
  return;

  readfile:;
  obsct=0;
  text=upcase(tag);
  input @1 @;
  do while(1);
    input @(trim(tag)) @;
    if c>&filesize then leave;
    col=c;
    obsct+1;
    output;
  end;
  return;
  run;
 proc sort;
 by col;
 run;
```

程序通过 link 语句实现 tag 不同值的搜索，link 将程序转向搜索程序段 readfile，搜索完成通过 return 语句返回。生成的 SAS 数据具有以下形式

	text	col
1	<TABLE	1734
2	<TR	1798
3	<TD	1805
4	<TABLE	1812
5	<TR	1887
6	<TD	1894
7	</TD	1973
8	</TR	1980
9	</TABLE	1990
10	</TD	1995

text 为表格标签，col 为该标签后的第一个字符所在的列。

call symput( )语句通过调用例程 symput( )将 data 步中关键数据赋值给宏变量，供下一个 data 步使用，实现 data 之间的数据传递。

step3：确定各个表格的行数和列数

在读取表格数据之前，需要根据搜寻到的表格标签及其位置确定各个表格中的行数和列数，为表格数据读取做好准备。

首先定义表格标签数组 taglist、标签起始列数组 tagstart 和结束列数组 tagend、表格起始列数组 tablestart 和结束列数组 tableend、表格行数数组 tablenrows 和列数数组 tablncols，然后将 step2 中生成的数据集中 text 变量值定义为数组元素以便引用，将标签所在列变量 col 值定义为数组元素以便引用。然后在 taglist 中寻找以'</'开头的标签，作为结束标签，往前寻找，如果遇到标签的第 3 个字符和当前 taglist 中第 4 个字符相同的，作为结束标签。例如：如果查询到 taglist='</TABLE'，则往回寻找 taglist 取值的第 3 个字符为'A'的标签，即标签值为'<TABLE'，其对应的 col 值即为表格结束的列数。以此确定表格结束的列数。程序段如下：

程序段 1：

```
    array taglist{&nobs} $8 _temporary_; /*标签数组*/
    array tagstart{&nobs} _temporary_; /*开始标签数组*/
    array tagend{&nobs} _temporary_; /*结束标签数组*/
    array tablestart{&ntables} _temporary_; /*表格开始位置数组*/
    array tableend{&ntables} _temporary_; /*表格结束位置数组*/
    array tablenrow{&ntables} _temporary_; /*表格行数数组*/
    array tablencols{&ntables} _temporary_; /*表格列数数组*/
/*把数据集tabeltag中变量text和col的值映射为数组taglist和tagstart元素*/
    do i=1 to &nobs.;
       set tabletag point=i;
       taglist{i}=text;
       tagstart{i}=col;
    end;
/*确定开始标签的位置和结束标签的位置*/
    data _null_;
     array taglist{&nobs} $8 _temporary_; /*标签数组*/
     array tagloc{&nobs} _temporary_;/*标签所在列*/
     array tagstart{&nobs} _temporary_; /*开始标签数组*/
     array tagend{&nobs} _temporary_; /*结束标签数组*/
     array tablestart{&ntables} _temporary_; /*表格开始位置数组*/
     array tableend{&ntables} _temporary_; /*表格结束位置数组*/
```

```
 array tablenrows{&ntables} _temporary_; /*表格行数数组*/
 array tablencols{&ntables} _temporary_; /*表格列数数组*/
 /*set tabletag;*/
 do i=1 to &nobs.;
   set tabletag point=i;
   taglist{i}=text;
   tagloc{i}=col;
 end;

 do i=1 to &nobs;
 if taglist{i}=:'</' then
    do j=i-1 to 1 by -1;
    if substr(taglist{j},3)=substr(taglist{i},4) then
      do;
      tagend{j}=tagloc{i};
      tagstart{j}=tagloc{j};
      leave;
      end;
    end;
 end;
/*确定表格开始标签的位置和结束标签的位置*/
j=0;
 do i=1 to &nobs;
 if taglist{i}='<TABLE' then
   do;
   j+1;
   tablestart{j}=tagstart{i};
   tableend{j}=tagend{i};
   end;
 end;
/*确定各个表格中包含的行数和列数*/
 do i=1 to &nobs;
  if taglist{i}='<TR' then
     do j=1 to &ntables;/*确定是哪个表格内的'<TR'标签*/
if tablestart{j}<=tagstart{i}<=tableend{j} then
     tablenrows{j}+1;/*表j行数加1*/
     end;
  if taglist{i}='<TD' then
     do j=1 to &ntables;/*确定是哪个表格内的'<TD'标签*/
       if tablestart{j}<=tagstart{i}<=tableend{j} then
       tablencols{j}+1;/*表j列数加1*/
     end;
end;
stop;
```

```
run;
```

step4：读取各个表格数据生成 SAS 数据集

确定了表格的个数、各个表格的起始和结束位置以及包含的行数和列数之后，可以对表格中数据进行读取。将读取表格数据程序段做成宏，并在循环中调用宏。

读取表格数据的宏%readtable

```
%macro readtable(tablenum,start,end,nrow,ncols);
data table1;
infile myfile recfm=f lrecl=&filesize. column=c missover;
array col{*} $200 col1-col&ncols.;
keep col1-col&ncols.;
/*将指针放置在表格开始位置*/
input @start& @; /*尾缀@保证data步内下一个input仍读取当前行*/
endrow=.;/*初始化行结束位置*/
/*读取每一行*/
do i=1 to &nrows;
  /*定位行的起始位置和结束位置：行以<TR或者<tr开头*/
input @"<&trtag"  @;/*定位指针到行开头*/
startcol=c;/*开始位置*/
  /*确定行到哪里结束，用<\tr标签，碰到下一个<tr,或者下一个<table*/
%if &closetags.=Y %then
  %do;
  input @"</&trtag" @;
  endrows=c-4;/*结束位置*/
  %end;
%else
  %do;
  if i<&nrows then
     do;
     input @"&trtag" @;
     endrow=c-4;
  else
     do;
     input "@&tabletag" @;
     endrow=c-7;/*碰到下一个表格开头确定行结束位置*/
     end;
  %end;
  /*从头读取行的内容*/
input @startcol @;/*定位指针到行的第一列*/
  /*读取改行所有列 */
  do j=1 to &ncols;
      /*列以<TD或者<td为开始标志*/
    input @"<&tdtag" @;
      /*碰到结束标志后将后续的列置为空白*/
```

```
    if c>=endrow then
      do k=j to &ncols;
      col{j}=' ';
      input @endrow @;
      leave;
      end;
       /*越过标签*/
      input @'<' @;/*指针移动到'<'之后*/
      /*计算列数据结束位置，用</td、<tr或者<table*/
    %if &closetags.=Y %then
       %do;input @"</&tdtag" @;%end;
    %else
       %do;
         if j<&ncols then input @"<&tdtag" @;
         else
         if i<&nrows then input @"<&trtag" @;
         else input @"<&tabletag" @;
       %end;
      /*读取指定范围内的所有内容*/
    l=c-5-startcol+1;
    input @startcol text $varying32767.l@;
    col{j}=text;
    end;
output;
end;
stop;
run;
%mend readtable;
```

在 data 步中调用表格读取宏，形成如下程序段：

程序段 2：

```
do i=1 to &ntables;
   if tablestart{i}>0 and tableend{i}>0
     and tablenrows{i}>0 and tablencols{i}>0 then
     do;
args=catx(',',i,tablestart{i},tableend{i}
                      ,tablenrows{i},tablencols{i});
%readtable(args);
    end;
end;
```

step5：完整的程序

```
data _null_;
 array taglist{&nobs} $8 _temporary_; /*标签数组*/
```

```
 array tagloc{&nobs} _temporary_;/*标签所在列*/
 array tagstart{&nobs} _temporary_; /*开始标签数组*/
 array tagend{&nobs} _temporary_; /*结束标签数组*/
 array tablestart{&ntables} _temporary_; /*表格开始位置数组*/
 array tableend{&ntables} _temporary_; /*表格结束位置数组*/
 array tablenrows{&ntables} _temporary_; /*表格行数数组*/
 array tablencols{&ntables} _temporary_; /*表格列数数组*/
 /*set tabletag;*/
 do i=1 to &nobs.;
   set tabletag point=i;
   taglist{i}=text;
   tagloc{i}=col;
 end;

 do i=1 to &nobs;
 if taglist{i}=:'</' then
    do j=i-1 to 1 by -1;
    if substr(taglist{j},3)=substr(taglist{i},4) then
      do;
      tagend{j}=tagloc{i};
      tagstart{j}=tagloc{j};
      leave;
      end;
    end;
 end;
/*确定表格开始标签的位置和结束标签的位置*/
j=0;
 do i=1 to &nobs;
 if taglist{i}='<TABLE' then
   do;
   j+1;
   tablestart{j}=tagstart{i};
   tableend{j}=tagend{i};
   end;
 end;
/*确定各个表格中包含的行数和列数*/
 do i=1 to &nobs;
  if taglist{i}='<TR' then
     do j=1 to &ntables;/*确定是哪个表格内的'<TR'标签*/
if tablestart{j}<=tagstart{i}<=tableend{j} then
     tablenrows{j}+1;/*表j行数加1*/
     end;
  if taglist{i}='TD' then
     do j=1 to &ntables;/*确定是哪个表格内的'<TD'标签*/
```

```
      if tablestart{j}<=tagstart{i}<=tableend{j} then
      tablencols{j}+1;/*表j列数加1*/
    end;
end;
do i=1 to &ntables;
   if tablestart{i}>0 and tableend{i}>0
     and tablenrows{i}>0 and tablencols{i}>0 then
     do;
args=catx(',',i,tablestart{i},tableend{i}
,tablenrows{i},tablencols{i});
%readtable(args);
    end;
end;
stop;
run;

%macro readtable(tablenum,start,end,nrow,ncols);
data table&tablenum.;
infile myfile recfm=f lrecl=&filesize. column=c missover;
array col{*} $200 col1-col&ncols.;
/*将指针放置在表格开始位置*/
input @start& @; /*尾缀@保证data步内下一个input仍读取当前行*/
endrow=.;/*初始化行结束位置*/
/*读取每一行*/
do i=1 to &nrows;
  /*定位行的起始位置和结束位置：行以<TR或者<tr开头*/
input @"<&trtag"  @;/*定位指针到行开头*/
startcol=c;/*开始位置*/
  /*确定行到哪里结束，用<\tr标签，碰到下一个<tr,或者下一个<table*/
%if &closetags.=Y %then
  %do;
  input @"</&trtag" @;
  endrows=c-4;/*结束位置*/
  %end;
%else
  %do;
  if i<&nrows then
     do;
     input @"&trtag" @;
     endrow=c-4;
  else
     do;
     input "@&tabletag" @;
     endrow=c-7;/*碰到下一个表格开头确定行结束位置*/
```

```
      end;
    %end;
    /*从头读取行的内容*/
  input @startcol @;/*定位指针到行的第一列*/
    /*读取改行所有列 */
    do j=1 to &ncols;
       /*列以<TD或者<td为开始标志*/
      input @"<&tdtag" @;
       /*碰到结束标志后将后续的列置为空白*/
      if c>=endrow then
       do k=j to &ncols;
       col{j}=' ';
       input @endrow @;
       leave;
       end;
        /*越过标签*/
       input @'<' @;/*指针移动到'<'之后*/
       /*计算列数据结束位置，用</td、<tr或者<table*/
      %if &closetags.=Y %then
         %do;input @"</&tdtag" @;%end;
      %else
         %do;
           if j<&ncols then input @"<&tdtag" @;
           else
           if i<&nrows then input @"<&trtag" @;
           else input @"<&tabletag" @;
         %end;
        /*读取指定范围内的所有内容*/
      l=c-5-startcol+1;
      input @startcol text $varying32767.l@;
      col{j}=text;
      end;
  output;
  end;
  stop;
  run;
  %mend readtable;
```

总结讨论:

外部文件作为一行读取：

当用 data 步把网页读取为外部文本文件（.txt）之后，采用 input 语句读取有关信息是，将整个外部文件作为一行反复读取十分方便。要实现这一目标，需要在 input 语句的外部文件引导语句 infile 中添加选项 recfm=f 和 lrecl=，选项 recfm=f 要求 input 语句以字

符长度识别行，其它行结束标志失效。lrecl=则给定数据行的长度，如果将上一个 data 提前得到的外部文件字符数 filesize 作为数据行长度 lrecl=filesize，则可以实现 input 语句将整个文件作为一行读取。

指针定位：

反复读取一行数据，寻找需要的信息，需要精准定位数据读取指针（pointer）。input 语句指针定位选项@n 和@Char 可以将指针定位在指定列和指定字符滞后的列，选项 column=c 将当前指针所在的列位置保存在临时变量 c 中。程序中往往先用语句 input@Char @；将指针定位到特定字符之后，然后用 input 读取行数据，尾缀@保证 data 步内下一个 input 语句仍然从当前行读取，因为只有一行数据，要反复读取需要尾缀@。

思考题:

能将本案例提供的方法用于文本挖掘吗？如果挖掘的是中文网页，如何处理中文字符的字节长度与英文不一致的问题?

案例 5：基于日内高频交易数据的价格波动计算[②]

案例背景：

波动（volatility）在金融理论和实践起着重要作用。连续时间金融认为，资产价格的波动具有随机性和时变性，是随机过程。波动过程在时间点上的值称为时点波动（spot volatility)。金融学中用方差计量波动，方差的不可观测性使波动成为隐变量(latent variable)，需通过资产价格样本间接进行估计。在较低频率下（例如周、月等），可采用 ARCH 类模型和 SV 模型等参数模型研究资产价格的波动，并采用极大似然方法估计模型参数。

近年来，大量高频算法交易的盛行，推动了金融学和金融计量经济学对高频数据的研究，高频时点波动的估计和推断显现出重要性。鉴于时点波动不可观测的特点，人们尝试采用非参数方法估计时点波动并将估计值作为“样本”，使波动过程将变得“可观测”，据此直接对波动进行分析或者以此作为进一步分析的基础，以克服波动的隐变量限制。

在高频交易数据中，资产价格经常发生跳跃（Jump）。跳的存在严重影响波动的估计准确性，包括积分波动（Integrated Volatility）和时点波动。文献中采用双幂变差（Bipower variation）和门限二阶变差（Threshold Quadratic Variation）两种方法消除跳的影响。本案例将消除跳影响的两种方法结合起来，采用门限双幂变差构造时点波动估计量，以减少跳漏滤带来的估计偏误。

问题归类:

1）采用矩阵编程语言 SAS/IML 进行模拟，计算时点波动；
2）综合运用 SAS/IML 和 BASE/SAS 中 data 步对实证数据进行整理；

教学过程:

step1.理论基础：

资产价格模型的设定包括收益方程和波动方程。设时刻t处的资产价格为P_t，对数价格（以下简称价格）为$p_t=\ln P_t$，其微分dp_t为收益（率）。Back (1991)证明，无套利市场价格过程p_t是特殊半鞅，服从跳扩散过程。时刻t处资产价格的波动用对数价格的方差σ_t^2来计量，称为时点波动。为保证波动的非负性，文献中常采平方根过程（也称 CIR 模型）刻画时点波动。为此，将资产价格模型设定为

$$dp_t=\mu_t dt+\sigma_t dw_t+dJ_t \qquad t\in[0,T]$$

，其中μ_t为适应过程（adapted process），w_t为标准布朗运动，表示收益模型风险源。$\mu_t dt+\sigma_t dw_t$为价格的连续部分，dJ_t为跳部分，文献中常常将J_t设定为复合 Poisson 过程，即$J_t=\sum_{i=1}^{N_t}\eta_i$，$N_t$为 Poisson 计数过程，跳发生密度为$\lambda_t$，$\eta_i$为第$i$个跳的幅度（size）。

为方便，设$T=1$。设$[0,1]$内等间隔时点$0=t_0<t_1<\cdots<t_n=1$处观测到的交易价格样本数据为$p_{t_i},i=0,2,\cdots,n$。$\delta=t_{i+1}-t_i=1/n$表示时间间隔。$\Delta_i p\equiv p_{t_i}-p_{t_{i-1}}$表示区间$[t_i,t_{i-1}]$上的价格改变。$(\Delta_i p)^2$称为价格过程$p_t$在$[t_i,t_{i-1}]$上的二阶变差，$0.5\pi|\Delta_i p||\Delta_{i+1}p|$称为双幂变差。二阶变差和双幂变差都是$p_t$在$[t_i,t_{i-1}]$积分波动的无偏

[②] 沈根祥. 基于门限双幂变差的时点波动非参数估计。中国管理科学 2015.24（1） p:21-29.

估计。$\Delta_i w = w_{t_i} - w_{t_{i-1}}$表示布朗运动$[t_i, t_{i-1}]$的改变量。为简化符号，用$\sigma_i^2$表示时点$t_i$处时点波动，即$\sigma_i^2 \equiv \sigma_{t_i}^2$。本文中的估计量基于核平滑非参数方法，对核函数和带宽作如下假设:

K1. 核函数$K(\cdot)$为非负实值连续可导函数，且导数有界，并满足

$$\int_R K(z)dz = 1, \ \int_R zK(z)dz < \infty, \ \ \int_R K^m(z)dz \equiv K_m < \infty, \ m = 2,4$$

K2. 带宽（bandwidth）h满足：（i）$h \to 0$；（ii）当$n \to \infty$、$h \to 0$时，$nh \to \infty$，$nh^2 \log(1/h) \to 0$。

对任意时点$t \in (0,T)$，设$K_h(s) = h^{-1}K[(s-t)/h]$为平滑核。定义资产价格过程时点波动估计量

$$\hat{\sigma}_t^{*2} = \frac{\sum_{i=1}^n K_h(t_i)\frac{\pi}{2}|\Delta_i z||\Delta_{i+1} z|}{\sum_{i=1}^n K_h(t_i)\delta},$$

$$\Delta_i z = \begin{cases} \Delta_i p, & (\Delta_i p)^2 \le \vartheta(\delta) \\ \text{med}(|\Delta_{i\pm 2} p|, |\Delta_{i\pm 1} p|), & (\Delta_i p)^2 \ge \vartheta(\delta) \end{cases}$$

，其中 med 表示中位数。

step2.蒙特卡洛模拟

本部分通过随机模拟来评价时点方差估计量的效果，考察 Poisson 跳对时点波动估计的影响以及时点波动估计量的有限样本表现，涉及问题包括带宽选择和门限选择。

1. 模拟数据

设数据生成过程为带跳的随机波动模型

$$\begin{cases} dp_t = \mu dt + \sigma_t dw_t + dJ_t, \\ d\sigma_t^2 = \kappa(\theta - \sigma_t^2)dt + v\sigma_t dB_t \\ dw_t dB_t = \rho dt \end{cases} \qquad t \in [0,T] \qquad (11)$$

，收益模型为包含 Poisson 跳的跳扩散过程，由于漂移项对时点波动估计量的渐进性质没有影响，故设$\mu_t = 0$。波动模型采用平方根过程以保证σ_t^2取正值。杠杆效应通过价格过程布朗运动w和波动过程布朗运动B的相关性来刻画，相关系数为ρ。模拟数据分两个步骤得到：第一步从扩散随机波动模型产生连续的资产价格模拟数据；第二步从复合 Poisson 过程产生跳模拟数据并与扩散模型产生的模拟数据相加。模拟以沪深 300 指数日内五分钟交易为对象，时间跨度为一年，按 250 个交易日计算，样本量$n = 250 \times 4 \times 12 = 12000$，$T = 1$，$\delta = 1/n$。模拟各种情况下资产价格过程$p_t$的 100 个轨道样本，计算出对应的时点方差估计值，以此计算出 ISE 对估计效果进行评价。参数取值参考沈根祥(2012b)，即$\theta = 3.3 \times 10^{-2}$、$\kappa = 0.1$、$v = 0.05$，为考察杠杆效应和跳发生频繁程度对估计结果的影响，在其他参数取值不变的情况下，取衡量杠杆效应强度的参数ρ的值分别为-0.25、-0.5 和-0.75、衡量跳发生频繁程度的参数λ的值分别取为 1/24、1/48、1/144 和 1/240，对各种情况模拟一条样本轨道。初始值σ_0^2从正态分布$N(5.3 \times 10^{-2}, 1.65 \times 10^{-2})$抽样产生。

SAS 程序代码为

```
proc iml;
/*预先定义用到的矩阵和参数*/
```

```
n=12000;/*样本量*/
rho=-.5;/*杠杆效应参数*/
lmd=1/144;/*跳发生频率参数*/
obs=(1:n)`;/*指标变量*/
delt=1/n;
delt_r=sqrt(delt);
t=1/(n**0.99);
b1=j(n,1,0);
b2=b1;

sig=j(n,1,0);/*时点波动序列*/
r=j(n,1,0);/*收益序列向量*/
jup=j(n,1,0); /*泊松跳序列*/
djup=j(n,1,0);/*泊松跳过程独立增量序列*/
rj=j(n,1,0);/*从收益序列*/

/*模拟100条路径*/
do l=1 to 100;
/*第一步:产生两个正态分布序列b1和b2;
据此参数两个相关序列w1和w2，相关系数为rho*/
do i=1 to n;
b1[i]=normal(238976);
b2[i]=normal(987652);
end;
w1=rho#b1+(sqrt(1-rho**2))#b2;
w2=b1;

/*第二步：从模型
sig2=0.1(0.053-sig2(-1))delta+0.002sqrt(sig2(-
1))*sqrt(delta)w2+sig2(-1)
中产生波动序列sig，从 N(0.0053,0.00011)抽样获得初始值*/

sig[1]=5.3e-2+normal(235465)*sqrt(1.1e-3);
do i=1 to n-1;
sig[i+1]=0.1*(0.033-
sig[i])*delt+0.05*sqrt(abs(sig[i]))*delt_r*w2[i]+sig[i];
end;

/*第三步:从离散化后的扩散过程r=sig[i-1]*sqrt(delt)*w1
        中抽取轨道样本*/

do i=2 to n;
r[i]=sqrt(abs(sig[i-1]))*delt_r*w1[i];
end;
```

```
/*第四步：从泊松计数过程N(t)中抽取泊跳样本j，（日内）跳发生概率为lmd=4
跳的大小服从分布ln(1+j)~N(ln(1+mu-1/2sigj2 sigj2))*/

m=n*lmd;  /*mean of number of jumps*/
l=ranpoi(27368,m);/*random number of jumps*/;
jt=j(l,1,0); /*jump times*/
do i=1 to l;
jt[i]=uniform(0);
end;
js=j(l,1,0); /*jump sizes*/
do i=1 to l;
z=log(1.08-0.0025/2)+0.05*rannor(547678);
js[i]=exp(z)-1;
end;

do i=1 to n;
jup[i]=sum(js#(jt<i*delt));
end;

/*第五步：生成跳扩散收益序列*/

do i=2 to n;
djup[i]=jup[i]-jup[i-1];
end;
rj=r+djup;/*return from jump-diffusion */
```

知识点：

（1）随机过程离散化是其模拟的重要一步，要掌握常用的离散化方法，将随机过程编程时间序列模型，然后进行模拟。

（2）随机数发生器的使用：模拟最为重要的算法是从特定分布中随机数。本案例的模拟中涉及到的分布包括：正态、泊松等。需要注意的是，既可以采用BAS/SAS中的随机数函数从特定分布中抽样，也可以采用SAS/IML中的随机数函数进行抽样，前者每次抽取一个样本，后者抽取给定矩阵下每个元素的样本，后者的抽样效率更高。

2. 时点波动估计

从时点波动估计式

$$\hat{\sigma}_t^{*2}=\frac{\sum_{i=1}^{n}K_h(t_i)\frac{\pi}{2}|\Delta_i z||\Delta_{i+1}z|}{\sum_{i=1}^{n}K_h(t_i)\delta},$$

$$\Delta_i z=\begin{cases}\Delta_i p, & (\Delta_i p)^2\le \vartheta(\delta)\\ \mathrm{med}(|\Delta_{i\pm2}p|,|\Delta_{i\pm1}p|), & (\Delta_i p)^2\ge \vartheta(\delta)\end{cases}$$

med 表示中位数

看出，估计时点波动$\hat{\sigma}_t^{*2}$，首先需要计算截尾收益$\Delta_i z$，然后计算交错相乘项$|\Delta_i z||\Delta_{i+1}z|$(Bipower)，最后计算时点波动估计。

程序代码（续）

```
r2=r##2;
r_r_1=r#lag(r);
rj=r+djup;/*return from jump-diffusion */
rj2=rj##2;/*suared return*/
rj2_t=rj2#(rj2<t);/*truncted squared return*/
rj_t=rj#(rj2<t);

/*用最邻近收益的均值和中位数取代门限过滤掉的收益*/

rja_2=j(n,1,0);
rjm_2=j(n,1,0);
do i=3 to n-2;
a=rj2_t[(i-2):(i+2)];
num=sum(a^=0);
a=sum(a)/num;
rja_2[i]=rj2[i]*(rj2[i]<=t)+(rj2[i]>t)*a;
call qntl(q,a,0.5);
rjm_2[i]=rj2[i]*(rj2[i]<=t)+(rj2[i]>t)*q;
end;
rja=sqrt(rja_2);
rjm=sqrt(rjm_2);
rj_rj_1a=rja#lag(rja,1);
rj_rj_1m=rjm#lag(rjm,1);

/*补充门限过滤掉收益的 ,双幂变差和平方收益序列*/
rjl=rj#(rj2<t)+r#(rj2>=t);/*jump returns not be filtered out*/;
rjca=r#(r2<t)+rja#(r2>=t);/*diffusion returns filtered out and
replaced by average of near returns*/;
rjcm=r#(r2<t)+rjm#(r2>=t);/*diffusion returns filtered out and
replaced by average of near returns*/;
rjl_2=rjl##2;
rjca_2=rjca##2;
rjcm_2=rjcm##2;
rjl_rjl_1=rjl#lag(rjl);
rjca_rjca_1=rjca#lag(rjca);
rjcm_rjcm_1=rjcm#lag(rjcm);

/*计算估计sig_2需要的带宽参数，门限值为t=sig1*9*/

  /*volatility estimation using truncated jump-diffusion returns*/
   sigpa_rj=j(n,1,0);
   sigpm_rj=j(n,1,0);
   sigbpa_rj=j(n,1,0);
```

```
   sigbpm_rj=j(n,1,0);
  /*volatility estimation using diffusion returns*/
   sigp_r=j(n,1,0);
   sigbp_r=j(n,1,0);
  /*volatility estimation using returns not filtered out jump
returns*/
   sigp_rjl=j(n,1,0);
   sigbp_rjl=j(n,1,0);
  /*volatility estimation using returns filtered out wrongly
continuous returns*/
   sigpa_rjc=j(n,1,0);
   sigpm_rjc=j(n,1,0);
   sigbpa_rjc=j(n,1,0);
   sigbpm_rjc=j(n,1,0);

c=0.4; /*bandwidth scale*/
c1=(c**2)*n;
uper=c*sqrt(n);/*working bandwidth*/
/*compute sig_1 and sig4_1 using Beta kernel K(x)=3/4(1-
x^2)I{|x|<=1}*/
do i=1 to n;
a2=(abs(obs-i)<=uper);
b=a2#(obs-i);
k=.75*(1-(b##2)/c1)#a2; /*Beta kernel vector*/
sumk=sum(k);

sigpa_rj[i]=sum(k#rja_2)/sumk;
sigpm_rj[i]=sum(k#rjm_2)/sumk;
sigbpa_rj[i]=sum(k#rj_rj_1a)/sumk;
sigbpm_rj[i]=sum(k#rj_rj_1m)/sumk;

sigp_r[i]=sum(k#r2)/sumk;
sigbp_r[i]=sum(k#abs(r_r_1))/sumk;

sigp_rjl[i]=sum(k#rjl_2)/sumk;
sigbp_rjl[i]=sum(k#abs(rjl_rjl_1))/sumk;

sigpa_rjc[i]=sum(k#rjca_2)/sumk;
sigpm_rjc[i]=sum(k#rjcm_2)/sumk;
sigbpa_rjc[i]=sum(k#abs(rjca_rjca_1))/sumk;
sigbpm_rjc[i]=sum(k#abs(rjcm_rjcm_1))/sumk;
end;
```

知识点：

（1）矩阵逐元素相乘：矩阵逐元素相乘运算“#”给有关计算带来极大方便。

（2）元素截尾：采用矩阵逻辑运算“>=”、“>”、“<=”、“<”等产生的选择矩阵，结合逐元素相乘能够实现元素截尾。例如 rja#(r2>=t)，r2>=t 产生的矩阵是元素为 0，1 的矩阵，比较结果为真（假）对应的元素为 1（0），与矩阵 rja 相乘后对 rja 的元素进行选择：条件成立的对应元素保留，不成立的对应元素值为 0.

3．模拟和估计宏

将以上各分解程序段写为宏（macro），实现 100 条模拟轨道的抽样。

模拟宏程序代码

```
%macro spotvol;
%do l=1 %to 100;
rj=r+djup;/*return from jump-diffusion */
rj2=rj##2;/*suared return*/
rj2_t=rj2#(rj2<t);/*truncted squared return*/
rj_t=rj#(rj2<t);

/*replace the ruturn filtered out by the average or median of
nearest returns*/

rja_2=j(n,1,0);
rjm_2=j(n,1,0);
do i=3 to n-2;
a=rj2_t[(i-2):(i+2)];
num=sum(a^=0);
a=sum(a)/num;
rja_2[i]=rj2[i]*(rj2[i]<=t)+(rj2[i]>t)*a;
call qntl(q,a,0.5);
rjm_2[i]=rj2[i]*(rj2[i]<=t)+(rj2[i]>t)*q;
end;
rja=sqrt(rja_2);
rjm=sqrt(rjm_2);
rj_rj_1a=rja#lag(rja,1);
rj_rj_1m=rjm#lag(rjm,1);

/*return ,bipowered and squared return of diffusion returs
fitered out and replaced by something
 return,bipowered and squared return of jump returns not be
filtered out*/
rjl=rj#(rj2<t)+r#(rj2>=t);/*jump returns not be filtered out*/;
rjca=r#(r2<t)+rja#(r2>=t);/*diffusion returns filtered out and
replaced by average of near returns*/;
rjcm=r#(r2<t)+rjm#(r2>=t);/*diffusion returns filtered out and
replaced by average of near returns*/;
rjl_2=rjl##2;
rjca_2=rjca##2;
rjcm_2=rjcm##2;
```

```
rjl_rjl_1=rjl#lag(rjl);
rjca_rjca_1=rjca#lag(rjca);
rjcm_rjcm_1=rjcm#lag(rjcm);

/*computation of bandwidth for estimation of sig_2 using
threshold t=sig1*9 */

  /*volatility estimation using truncated jump-diffusion returns*/
   sigpa_rj=j(n,1,0);
   sigpm_rj=j(n,1,0);
   sigbpa_rj=j(n,1,0);
   sigbpm_rj=j(n,1,0);
  /*volatility estimation using diffusion returns*/
   sigp_r=j(n,1,0);
   sigbp_r=j(n,1,0);
  /*volatility estimation using returns not filtered out jump
returns*/
   sigp_rjl=j(n,1,0);
   sigbp_rjl=j(n,1,0);
  /*volatility estimation using returns filtered out wrongly
continuous returns*/
   sigpa_rjc=j(n,1,0);
   sigpm_rjc=j(n,1,0);
   sigbpa_rjc=j(n,1,0);
   sigbpm_rjc=j(n,1,0);

c=0.4; /*bandwidth scale*/
c1=(c**2)*n;
uper=c*sqrt(n);/*working bandwidth*/
/*compute sig_1 and sig4_1 using Beta kernel K(x)=3/4(1-
x^2)I{|x|<=1}*/
do i=1 to n;
a2=(abs(obs-i)<=uper);
b=a2#(obs-i);
k=.75*(1-(b##2)/c1)#a2; /*Beta kernel vector*/
sumk=sum(k);

sigpa_rj[i]=sum(k#rja_2)/sumk;
sigpm_rj[i]=sum(k#rjm_2)/sumk;
sigbpa_rj[i]=sum(k#rj_rj_1a)/sumk;
sigbpm_rj[i]=sum(k#rj_rj_1m)/sumk;

sigp_r[i]=sum(k#r2)/sumk;
sigbp_r[i]=sum(k#abs(r_r_1))/sumk;
```

```
sigp_rjl[i]=sum(k#rjl_2)/sumk;
sigbp_rjl[i]=sum(k#abs(rjl_rjl_1))/sumk;

sigpa_rjc[i]=sum(k#rjca_2)/sumk;
sigpm_rjc[i]=sum(k#rjcm_2)/sumk;
sigbpa_rjc[i]=sum(k#abs(rjca_rjca_1))/sumk;
sigbpm_rjc[i]=sum(k#abs(rjcm_rjcm_1))/sumk;
end;

r&l=r;
rj&l=rj;
rj_t&l=rj_t;
jup&l=jup;

rjl&l=rjl;
rjca&l=rjca;
rjcm&l=rjcm;

sig&l=sig;
sigpa_rj&l=n*sigpa_rj;
sigpm_rj&l=n*sigpm_rj;
sigbpa_rj&l=(3.1416/2)*n*sigbpa_rj;
sigbpm_rj&l=(3.1416/2)*n*sigbpm_rj;

sigp_r&l=n*sigp_r;
sigbp_r&l=(3.1416/2)*n*sigbp_r;

sigp_rjl&l=n*sigp_rjl;
sigbp_rjl&l=(3.1416/2)*n*sigbp_rjl;

sigpa_rjc&l=n*sigpa_rjc;
sigpm_rjc&l=n*sigpm_rjc;
sigbpa_rjc&l=(3.1416/2)*n*sigbpa_rjc;
sigbpm_rjc&l=(3.1416/2)*n*sigbpm_rjc;
%end;
%mend;
```

4．将模拟结果输出为 SAS 数据集

为比较时点波动估计效果，需要先将模拟结果输出为 SAS 数据集。

程序代码

```
%spotvol;
%macro name;
%do l=1 %to 100;
r&l
```

```
rj&l
rj_t&l
jup&l
rjl&l
rjca&l
rjcm&l

sig&l
sigpa_rj&l
sigpm_rj&l
sigbpa_rj&l
sigbpm_rj&l

sigp_r&l
sigbp_r&l

sigp_rjl&l
sigbp_rjl&l

sigpa_rjc&l
sigpm_rjc&l
sigbpa_rjc&l
sigbpm_rjc&l
%end;
%mend;
create est_rho50_lmd144 var{obs %name};
append;
quit;
```

知识点：

（1）用宏生成变量名：由于模拟的收益序列和波动序列多达 100*20=2000 个，需要给每个序列命名。采用宏循环能够避免代码的重复，极大简化工作量和程序长度。

（2）在 create 命令的 var 从句中用宏运行%name，确定输出数据集的观测变量名。

step4. 估计效果评价——偏误比较

为了对时点波动估计量跳漏滤偏误进行比较，对参数取不同值的每一种情况各模拟 100 条样本轨道，计算 $\hat{\sigma}^{*2}$ 和 $\hat{\sigma}_s^2$ 的单个路径累积偏误均值和平均路径累积偏误均值，并进行比较。单个路径累积偏误均值（Bias1）和平均路径累积偏误（Bias2）定义为

$$\text{Bias1}=\frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{T}|\hat{\sigma}_{it}^{(m)2}-\hat{\sigma}_{it}^{2}|,\qquad \text{Bias2}=\sum_{t=1}^{T}\left|\frac{1}{N}\sum_{n=1}^{N}\hat{\sigma}_{it}^{(m)2}-\frac{1}{N}\sum_{n=1}^{N}\hat{\sigma}_{it}^{2}\right|$$

，其中 $\hat{\sigma}_{it}^{(m)2}$ 是采用跳漏滤数据 $\Delta p_{it}^{(m)}$ 由（8）计算的时点波动估计值，$\hat{\sigma}_{it}^{2}$ 是采用无漏滤数据 $\Delta p_{it}^{(c)}$ 计算的时点波动估计值，下标 it 表示第 i 条模拟轨道的 t 时点处。$\Delta p_{it}^{(m)}$ 定义为

$$\Delta p_{it}^{(m)} = \Delta p_{it}^{(d)} I_{\{(\Delta_{it} p)^2 < \vartheta(\delta)\}} + \Delta p_{it}^{(c)} I_{\{(\Delta_{it} p)^2 > \vartheta(\delta)\}}$$

Bias1和Bias2的计算采用BASE/SAS DATA步和proc means进行。首先用100个估计序列计算每个时点处波动的样本方差.

程序代码

```
data test1;
set est_rho50_lmd144;
   Var_sigpa_rj=var(of sigpa_rj1-sigpa_rj100);
   Var_sigpm_rj=var(of sigpm_rj1-sigpm_rj100);
   Var_sigbpa_rj=var(of sigbpa_rj1-sigbpa_rj100);
   Var_sigbpm_rj=var(of sigbpm_rj1-sigbpm_rj100);
run;
```

知识点：函数自变量名的缩略列示

函数自变量的缩略表示。每个var函数都有100个自变量，自变量前缀相同，采用缩略表示十分方便。需要注意：当函数自变量名用缩略列示时，要有自变量列表前要用of进行引导。

然后计算波动估计按轨道的累积方差

程序代码

```
proc means mean ;
var Var_sigbpa_rj Var_sigbpm_rj Var_sigpa_rj Var_sigpm_rj ;
title 'Ivar rho=-0.5 lmd=1/144';
run;
```

知识点：proc means和DATA步的区别

都是计算方差，横截面方差的计算采用了DATA步方差函数VAR()，而轨道累积方差的计算则采用proc means。其原因在于，横截面方差时100个序列对应变量的方差，适合用DATA步的跨列（跨变量）统计函数，而proc means计算的时同一变量跨行（跨观测）的方差。

最后计算出Bias1和Bias2

程序代码

```
data biassqred;
set est_rho50_lmd144;
   SqredBias_sigpa_rj=(mean(of sigpa_rj1-sigpa_rj100)-mean(of
sig1-sig100))**2;
   SqredBias_sigpm_rj=(mean(of sigpm_rj1-sigpm_rj100)-mean(of
sig1-sig100))**2;
   SqredBias_sigbpa_rj=(mean(of sigbpa_rj1-sigbpa_rj100)-mean(of
sig1-sig100))**2;
   SqredBias_sigbpm_rj=(mean(of sigbpm_rj1-sigbpm_rj100)-mean(of
sig1-sig100))**2;
   run;
proc means mean data=biassqred;
var SqredBias_sigpa_rj  SqredBias_sigpm_rj  SqredBias_sigbpa_rj
SqredBias_sigbpm_rj;
```

```
title 'IBias rho=-0.5 lmd=1/144';
run;
```

总结讨论:

SAS 各种产品的综合使用：

从本案例可看出，实际问题中的数据处理和分析往往需要综合使用 SAS 的各种产品。本案例在模拟阶段大量采用了 SAS/IML 的矩阵运算功能，极大提高了随机数参数的效率和灵活性。而 SAS/BASE 中 DATA 的使用使得跨变量的统计计算十分方便，过程步 Proc means 轻松实现了跨观测的统计计算。此外，宏的运用极大地减少了程序代码的重复，使得 100 条轨道的模拟通过宏循环实现。

此外，本案例告诉我们，要想有效应用软件处理实际为题，专业知识和技能十分重要。本案例计算的时点波动需要十分专业的数理金融和金融计量经济学知识。软件知识工具，只有对要出了的实际问题十分了解并给出执行方案后，才能有效利用软件。

思考题:

能否用本案例的随机过程模拟方法计算衍生产品定价？例如期权定价。能否将本案例的随机过程模拟方法用于风险价值VaR的计算？