8.7　AR(2)的 Bootstrap 算法详解：

1. 首先中心化z_t=$y_t - \bar{y}$　　AR(2)模型定义为$z_t = \beta_1 z_{t-1} + \beta_2 z_{t-2} + \varepsilon_t$，其中$\beta_1, \beta_2$未知需要我们后续求解.；t 为时间序列中的期数 t = U,U+1,⋯,V.(ps:在本题后续 demo 中所用的书上的数据，因此 t =3~48)，ε_t为噪声，E(ε_t) = 0,且 $\varepsilon_t = z_t - \widehat{\beta_1} z_{t-1} - \widehat{\beta_2} z_{t-2}$。由于$\varepsilon_t^*$服从经验分布$\hat{\mathrm{F}} \to (\varepsilon_3^*, \varepsilon_4^*, \dots, \varepsilon_{48}^*)$，而$\varepsilon_t^*$是$\hat{\mathrm{F}}$中进行不放回的简单随机抽样（或者书中采用的 Moving block）得到的，然后根据$z_t^* = \widehat{\beta_1} z_{t-2} + \widehat{\beta_2} z_{t-1} + \varepsilon_t^*$ 。
2. 根据抽样得出的样本，根据公式$\hat{\beta} = (Z^T Z)^{-1} Z^T z = \binom{\widehat{\beta_1}}{\widehat{\beta_2}}$ 得出未知参数的估计值，这个过程重复 B 次。每一次抽样得到一组($\widehat{\beta_1}^*, \widehat{\beta_2}^*$)，得到 B 组($\widehat{\beta_1}^*, \widehat{\beta_2}^*$)，然后可以计算出$\widehat{se}_{\beta_1}, \widehat{se}_{\beta_2}$。

。

In [3]:

```
# import numpy as np
import random
import numpy as np
import pandas as pd
from statsmodels.tsa.arima_model import ARIMA
import statsmodels.api as sm
```

习题8.1

由于在z,x独立，根据题目所给公式var(x̄-z̄)=var(x̄) + var(z̄),因此只要分别计算x，y的bootstrap的var(x̄star)，var(z̄star)，再将二者求和再开跟

In [4]:

```
x = np.array([94,197,16,38,99,141,23]) #treatment
z = np.array([52,104,146,10,51,30,40,27,46])#control
```

In [5]:

```
N = 1400
x_bar_bootstrap = []
z_bar_bootstrap = []
x_bootstrap = []
z_bootstrap = []
for i in range(N):
    x_random_choice = np.random.choice(x,len(x))  #从x中自助抽样 len(x)个
    x_bootstrap.append(x_random_choice)  #把这个值装进前面设置的抽样空列里
    x_mean = np.mean(x_random_choice)  #计算均值
    x_bar_bootstrap.append(x_mean)        #装进均值空列
    z_random_choice = np.random.choice(z,len(z))    #从z中自助抽样 len(Z)个
    z_bootstrap.append(z_random_choice)
    z_mean = np.mean(z_random_choice)
    z_bar_bootstrap.append(z_mean)
a = np.sqrt(np.var(x_bar_bootstrap) + np.var(z_bar_bootstrap)) #计算根号下(var(x̄star)+var(z̄star))
print("Se(theta) is :",a)     #One-sample bootstrap Se
```

```
Se(theta) is : 26.835389217929446
```

通过实验结果与书上Se(theta) = 26.86很接近。

In [7]:

```
x_bootstrap[:10]#展示前10个
```

Out[7]:

```
[array([38, 23, 23, 94, 38, 16, 99]),
 array([ 94, 197,  99,  23,  38, 141,  23]),
 array([ 23, 141,  23,  99,  94,  99, 141]),
 array([ 99, 141, 197,  38, 197,  99, 141]),
 array([ 38,  38, 141,  38,  23,  38,  16]),
 array([ 23, 197,  23, 141, 197,  23,  16]),
 array([ 23,  23,  38, 141, 141,  94,  38]),
 array([ 23,  23, 141,  16,  16, 141, 141]),
 array([141,  99, 197,  99, 141, 197,  16]),
 array([197,  94,  38,  23, 141,  94,  23])]
```

In [8]:

```
z_bootstrap[:10]#展示前10个
```

Out[8]:

```
[array([ 27,  40, 104,  46,  30,  51,  27,  27,  52]),
 array([ 27,  30, 104,  46, 146,  40,  51, 146, 104]),
 array([ 51,  46, 146,  51,  40,  51,  27,  51,  52]),
 array([146,  40,  52,  27,  10,  51, 146,  40, 104]),
 array([ 40, 146,  52, 104,  51,  40,  30, 104, 146]),
 array([ 52, 104,  51,  40,  30,  40,  40,  27,  51]),
 array([ 10,  10, 104, 146,  40,  40,  10,  46, 104]),
 array([ 10,  46, 104, 104, 104,  10,  40,  51,  27]),
 array([ 46,  52,  10,  27,  10,  30,  27, 146,  40]),
 array([ 46, 104,  52,  46,  40, 104, 146,  40,  52])]
```

因为我们前面是根据独立性，将*x*，*z*拆开来分别进行一次*one sample bootstrap*，所以 a题 *one sample bootstrap* 的是导出的是两个分别*shape*为*14007*和*14009*的矩阵，与*two-sample bootstrap shape*为*1400*16*的矩阵不一样。但通过比较*Se*值，可以发现二者非常接近

习题8.7 （详解部分见第一页pdf，本demo抽样方法用的是moving block）

Give a detailed description of the bootstrap algorithm for the second-order autoregressive scheme.

与书中一阶情况类似，我们使用moving block bootstar抽样，从时间序列中抽取连续的数据块组合成新的数据，然后对抽取的数据做建立2阶自回归模型，这个过程重复200次

以书上的激素水平在身体里时间序列数据为例

In [15]:

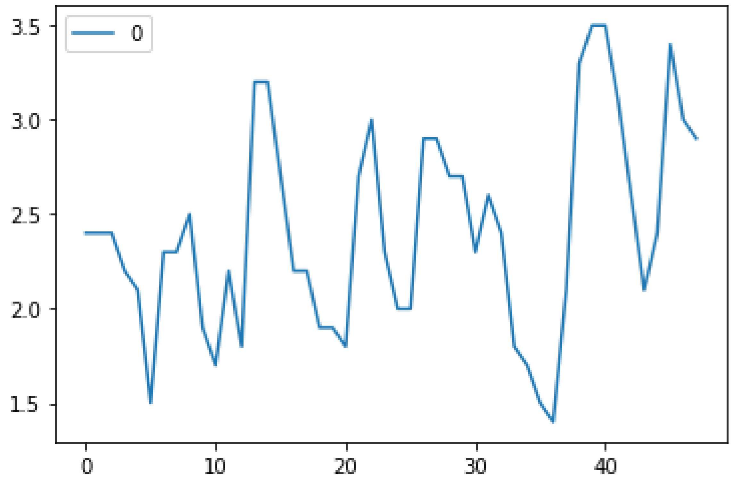
```
x = np.array([2.4,2.4,2.4,2.2,2.1,1.5,2.3,2.3,2.5,1.9,1.7,2.2,1.8,3.2,3.2,2.7,
              2.2,2.2,1.9,1.9,1.8,2.7,3.0,2.3,2.0,2.0,2.9,2.9,2.7,2.7,2.3,2.6,2.4,
              1.8,1.7,1.5,1.4,2.1,3.3,3.5,3.5,3.1,2.6,2.1,2.4,3.4,3.0,2.9])
```

In [16]:

```
pd.DataFrame(x).plot() #原始数据时序图
```

Out[16]:

<matplotlib.axes._subplots.AxesSubplot at 0x117b37b8af0>



In [17]:

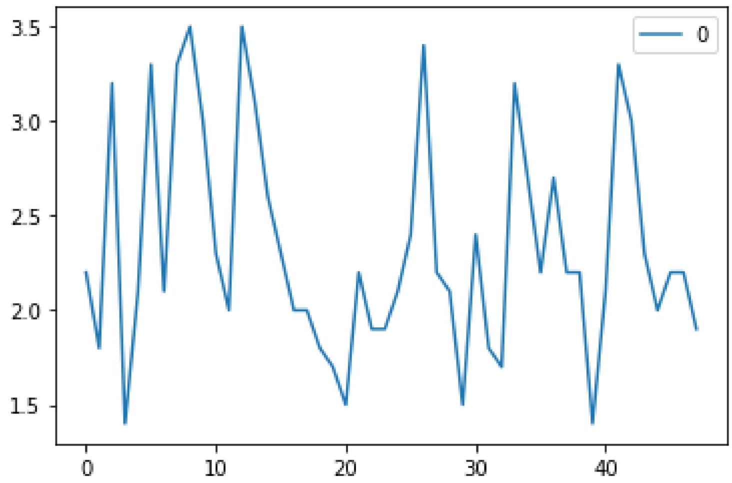
```
def move_block(i,n):  #设置 MOVING BLOCKS BOOTSTRAP 函数，在列表中选取三个连续位置的值
    x_take = x[i:i+n]
    #k = int(len(x)/n)
    return x_take#,k
```

In [18]:

```
move_block_data = []
k = int(len(x)/3)
for j in np.random.choice(range(len(x)-2),k):  #在1-46随机抽取下标
    x_3 = move_block(j,3)
    move_block_data.append(x_3)
a = np.array(move_block_data).reshape((48,))
pd.DataFrame(a).plot()
```

Out[18]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x117b3813220>
```



可见通过MOVING BLOCKS BOOTSTRAP产生时序图

做200次

In [20]:

```
k = int(len(x)/3)  #Move Block跨度为3
N = 200  #200次抽样

beta1 = []
beta2 = []
for i in range(N):
    data_bs = []
    for j in np.random.choice(range(len(x)-2),k):  #在1-46随机抽取下标
        x_3 = move_block(j,3)#调用Moving block函数
        data_bs.append(x_3)
    a = np.array(data_bs).reshape((48,))#展开成一维序列
    data =pd.DataFrame(a)
    model = ARIMA(data, (2,0,0)).fit() #AR(2)模型建立
    beta1.append(model.params[1])
    beta2.append(model.params[2])#model.params提取AR(2)模型参数
print("Se_Beta_1 estimate is:",np.std(beta1))
print("Se_Beta_2 estimate is:",np.std(beta2))
```

```
Se_Beta_1 estimate is: 0.14303660859164424
Se_Beta_2 estimate is: 0.13147156689738781
```

200次move block bootstap Ar(2）产生的值Beta1,Beta2

In [21]:

```
beta1[:10]#10 in 200 _Beta_1 estimate
```

Out[21]:

```
[0.5329808223153639,
 0.35749473657782016,
 0.3527246652572202,
 0.6578822616847326,
 0.1922020703112702,
 0.6172218144783868,
 0.293568226891683,
 0.42650185839334454,
 0.48715389815581445,
 0.3967523725434736]
```

In [23]:

```
beta2[:10] #10 in 200 _Beta_1 estimate
```

Out[23]:

```
[-0.2663729331453428,
 0.10414114851351565,
 -0.05112295355231362,
 -0.19989634260369135,
 -0.16207931361011596,
 -0.23629648339628304,
 -0.15351322432746695,
 -0.31573381610214496,
 -0.14352291683951876,
 -0.10606664482778762]
```