课程名称：《风险管理的数学方法》
提交日期：2019年9月24日
姓名：胡庆涛
学号：1901210003

1、债券组合风险刻画

某资产组合包含d只无风险债券，到期日分别为T_i, s时刻的价格分别为$p(s,T_i)$, 记到期日为T_i的债券个数为λ_i, $y(s,T)$表示s时刻到期日为T债券的到期收益率有$p(s,T)=e^{-(T-s)y(s,T)}$，用Δ转换时间刻度后，组合在时间t的价值为：

$$V_t=\sum_{i=1}^{d}\lambda_i p(t\Delta,T_i)=\sum_{i=1}^{d}\lambda_i exp(-(T_i-t\Delta)y(t\Delta,T_i))$$

记风险因子变化为$X_{t+1,i}=y((t+1)\Delta,T_i)-y(t\Delta,T_i)$ 故有，

$$\begin{aligned}L_{t+1}&=-(V_{t+1}-V_t)\\&=-(\sum_{i=1}^{d}\lambda_i p((t+1)\Delta,T_i)-\sum_{i=1}^{d}\lambda_i p(t\Delta,T_i)\\&=-\sum_{i=1}^{d}\lambda_i p(t\Delta,T_i)e^{y(t\Delta,T_i)\Delta-(T_i-t\Delta)X_{t+1,i}-1}\end{aligned}$$

由上式易知，线性损失为：

$$L_{t+1}^{\Delta}=-\sum_{i=1}^{d}\lambda_i p(t\Delta,T_i)(y(t\Delta,T_i)\Delta-(T_i-t\Delta)X_{t+1,i})$$

进一步近似，假定收益率曲线水平，即$y(s+\Delta)=y(s)+\delta$对所有T成立，可得

$$L_{t+1}^{\Delta}=-V_t(y_t\Delta-\underbrace{\sum_{i=1}^{d}\frac{\lambda_i p(t\Delta,T_i)}{V_t}(T_i-t\Delta)\delta)}_{Duration}$$

2、货币远期风险刻画

货币远期定义为：交易双方签订合约，在T时刻以约定汇率 $\bar{e}$ 买入（或者卖出）$\overline{V}$ 数量的外币。

因此，持有外币远期多头可以理解为T时刻持有外币的多头，以及一个到期率为$\bar{e}$的本币零息债券，e_T表示T时刻汇率（一单位外币可兑换本币的数量），故T时刻该组合以本币计算的价值为：

$$V_T=\overline{V}e_T+(-\overline{V}\bar{e})=\overline{V}(e_T-\bar{e})$$

对于国内零息债券的处理思路可参照第一部分，以下探讨持有外币零息债券头寸的处理思路，记$p^f(s,T)$为外币零息债券的价格， 则货币远期的风险因子为汇率以及外币零息债券的到期收益率，因子为$Z_t=(lne_t,y^f(s,T))'$, 故外币债券头寸部分的价值为：

$$V_t=\overline{V}e_t exp(-(T-t\Delta)y^f(t\Delta,T))=\overline{V}exp(Z_{t,1}-(T-t\Delta)Z_{t,2})$$

记风险因子变化为$X_{t+1,1}=lne_{t+1}-lne_t$ ；$X_{t+1,2}=y^f((t+1)\Delta,T)-y^f(t\Delta,T)$, 则有：

$$
\begin{aligned}
L_{t+1} &= -(V_{t+1} - V_t) \\
&= -\overline{V}\,exp(Z_{t,1} - (T - t\Delta)Z_{t,2})(e^{Z_{t,2}\Delta + X_{t+1,1} - (T-t\Delta)X_{t+1,2}} - 1)
\end{aligned}
$$

由上式易知，线性损失为：

$$L_{t+1}^{\Delta} = V_t(Z_{t,2}\Delta + X_{t+1,1} - (T - t\Delta)X_{t+1,2})$$

3、风险贷款组合的风险刻画

该例引入信用风险敞口，m个不同的交易对手，风险敞口的大小分别为 e_i ，引入示性随机变量来表示第 i 个对手在 $[0,t]$ 时间段内的违约状况，记作 $Y_{t,i}$ ， $Y_{t,i} = 1$ 表明发生违约，并且简单地假定为全部资产无法回收。

在收益率 $y(t,T)$ 后添加 $c_i(t,T)$ 来刻画第 i 个交易对手的信贷息差，故 t 时刻的价值表示为：

$$V_i = e_i e^{-(T-t)(y(t,T)+c_i(t,T))}$$

简化处理，假设 对所有交易对手而言， $c_i(t,T) = c(t,T)$，引入违约情形下，该资产组合在 t 时刻的价值为：

$$V_t = \sum_{i=1}^{m}(1 - Y_{t,i})exp(-(T-t)(y(t,T) + c(t,T)))e_i$$

风险因子为：

$$Z_t = (Y_{t,1}, \ldots, Y_{t,m}, y(t,T), c(t,T))^{'}$$

由于该例含离散项，故无法使用泰勒公式来进行一阶线性近似。

4、生成t分布随机数---C++

使用$Box—Muller$算法生成正态分布随机数，即$X = cos(2\pi U_1)\sqrt{-2lnU_2}$,$U_1$、$U_2$是$[0,1]$均匀分布的随机数，则X为标准正态分布的随机数。

证明如下：

X，Y 分别服从标准正态分布且相互独立，则联合概率密度为：

$f(x,y) = \frac{1}{\sqrt{2\pi}}e^{-\frac{x^2+y^2}{2}}$

将其进行极坐标变换，使$X = Rcos(\theta), Y = Rsin(\theta)$

可得 R 与 θ 的分布函数分别为：

$F_R(r) = \int_0^{2\pi}\int_0^r \frac{1}{2\pi}e^{-\frac{R^2}{2}}Rd\theta dR = 1 - e^{-\frac{r^2}{2}}$

$F_\Theta(\theta) = \int_0^{\theta}\int_0^{\infty}\frac{1}{2\pi}e^{-\frac{R^2}{2}}Rd\theta dR = \frac{\theta}{2\pi}$

而$F_R(r)$反函数为$R = \sqrt{-2ln(1-Z)}$,且Z服从$[0,1]$均匀分布时，R分布函数为上式

故令$\theta = 2\pi U_1$， $R = \sqrt{-2lnU_2}$代入可得 X 与 Y 的表达式.

生成 t 分布的思路为利用标准正态分布和卡方分布。代码如下：

```
//生成t(n)分布的随机数
#include <iostream>
#include <random>
#include <cmath>
#include <fstream>
#include<time.h>
using namespace std;
double pi = 3.1415926897932384;

//生成Chi_Square(n)分布的随机数
```

```cpp
double Chisquare(int n) {
    double z = 0;
    for (int i = 1; i <= n; i++) {
        double u1 = rand() % RAND_MAX / (double)RAND_MAX;
        double u2 = rand() % RAND_MAX / (double)RAND_MAX;
        double y = sqrt(-2 * log(u1)) * (cos(2 * pi * u2));//y为标准正态分布的随机数
        z += y * y; //运算得到chi_Square（n）分布随机数
    }
    return z;
}

int main() {
    int n;
    cout << "t分布自由度为: " << endl;
    cin >> n;
    ofstream fout("C:\\Users\\Lenovo\\Desktop\\tRandom.txt");

    for (int i = 1; i <= 10000; ++i) {
        double u1, u2, x;
        srand(i); // 每次设置不同的随机数seed

        //使用Box-Muller变换生成正态分布随机数
        u1 = rand() % RAND_MAX / (double)RAND_MAX; //生成（0，1）的随机数
        u2 = rand() % RAND_MAX / (double)RAND_MAX;
        x = sqrt(-2 * log(u1)) * sin(2 * pi * u2); //x为标准正态分布的随机数

        double z = Chisquare(n);
        double t = x / sqrt(z / n);

        //如果结果是inf或者nan就再进行一遍
        if (isinf(t) || isnan(t)) {
            i--;
            continue;
        }
        fout << t << endl;//将生成的随机数储存到文件内，以备检验
    }
    fout.close();
}
```

对生成的随机数进行检验，运用 $python$ 绘制频率直方图，然后与 t 分布进行比较，代码如下：

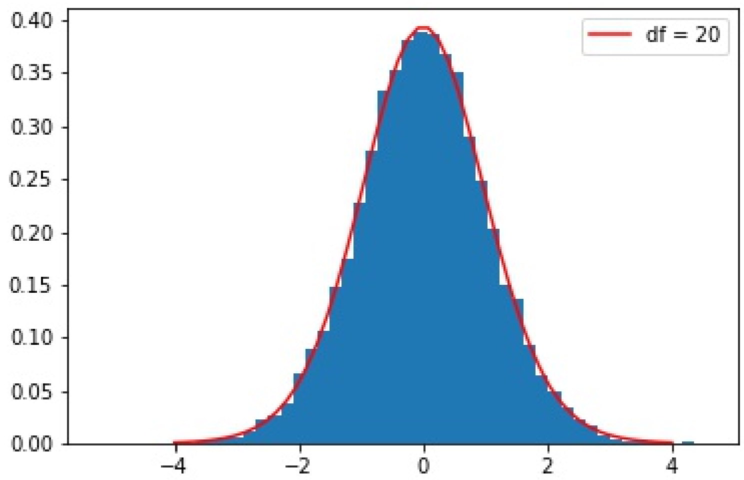
```python
import matplotlib.pyplot as plt
import numpy as np
from scipy.stats import t

if __name__ == '__main__':
    with open(r'C:\Users\Lenovo\Desktop\tRandom.txt','r') as f: #读取储存的数据
        random =[]
        for line in f:
            random.append(float(line.strip()))
        n = 20 #此处自由度应与生成随机数时所指定的相同
        c = 'df = '+ str(n)
        x = np.linspace(-4,4,50)
        plt.plot(x, t.pdf(x,n),label=c,color ='r') #绘制t分布的图像

        #绘制随机数的频率直方图，以进行比较
        plt.hist(random,bins=50,density = True)
```

```
        plt.legend()
        plt.savefig('trandom.jpg')
    exit(0)
```

生成的图片如下，证明随机数为 $t(20)$ 分布的随机数。



5、近似求解Expected Shortfall---python

针对正态分布，使用近似求解，并将n逐渐变大时的结果与精确解比较，验证近似效果。

正态分布精确求解公式为：$ES_\alpha = \mu + \sigma\frac{\phi(\phi(1-\alpha))}{1-\alpha}$，该公式证明如下：

$X \sim N（\mu, \sigma^2)$ 其反函数为 $\mu + \sigma\Phi^{-1}(s)$

$$
\begin{aligned}
ES_\alpha &= \frac{1}{1-\alpha}\int_\alpha^1 q_u(F_L)du \\
&= \frac{1}{1-\alpha}\int_\alpha^1 (\mu + \sigma\Phi^{-1}(s))ds \\
&= \mu + \sigma\underbrace{\frac{\int_\alpha^1 \Phi^{-1}(s)ds}{1-\alpha}}
\end{aligned}
$$

花括号内积分可以写作：

$$
\begin{aligned}
\int_\alpha^1 \Phi^{-1}(s)ds &= \int_{\Phi^{-1}（\alpha)}^{\infty} s\frac{e^{\frac{s^2}{2}}}{\sqrt{2\pi}}ds \\
&= \Phi(\Phi^{-1}(\alpha))
\end{aligned}
$$

代码如下：

```
import numpy as np
import scipy.stats as stats
import matplotlib.pyplot as plt

#定义ES近似计算公式
def func(n):
    l = np.random.normal(mu, sigma, size=n) #生成正态分布随机数
    L = sorted(l, reverse=True) #对随机数进行降序排序
    k = int(n*(1-alpha))
    if k >1: #k大于1时进行求和运算才有意义
        s = np.sum(L[0:k-1])/k
```

```
        return s
    return 0

#代入元素值进行计算
mu = 0
sigma =2
alpha = 0.99

#列出不同n对应的近似值
es = []
for n in range(1,int(1e5)):
    est = func(n)
    es.append(est)

#精确计算正态分布ES的值
ES = mu + sigma * stats.norm.pdf(stats.norm.ppf(alpha))/(1-alpha)

#作图查看n变大时的近似效果
plt.plot(es)
plt.axhline(y = ES, color='r',linestyle='-')
plt.show()
```

结果生成的图片如下，可见近似效果明显：