Rezerwa składki netto

June 11, 2022

Celem prezentowanego projektu jest obliczenia rezerw składek netto dla każdego roku trwania ubezpieczenia na życie i dożycie. Importujemy pakiet pandas, który będzie potrzebny do utworzenia tabeli.

[1]:
```
import pandas as pd
```

Następnie wczytywane są dane z tablic życia dla kobiet.

[2]:
```
lx = [100000,
99647,
99627,
99612,
99600,
99590,
99581,
99574,
99566,
99559,
99551,
99541,
99530,
99518,
99504,
99489,
99472,
99452,
99429,
99402,
99374,
99346,
99320,
99295,
99272,
99249,
99224,
99198,
99171,
99142,
```

```
99112,
99079,
99045,
99007,
98965,
98920,
98870,
98815,
98754,
98686,
98613,
98532,
98444,
98345,
98234,
98110,
97970,
97813,
97639,
97447,
97234,
97001,
96744,
96462,
96150,
95804,
95416,
94981,
94493,
93949,
93349,
92693,
91985,
91224,
90410,
89537,
88598,
87585,
86492,
85311,
84039,
82670,
81202,
79629,
77945,
76142,
74214,
```

```
72150,
69934,
67543,
64943,
62098,
58980,
55577,
51912,
48039,
44031,
39965,
35903,
31900,
27997,
24238,
20663,
17325,
14265,
11519,
9108,
7040,
5310,
3902,
2788]
qx=[0.00353,
0.00020,
0.00016,
0.00012,
0.00010,
0.00008,
0.00008,
0.00007,
0.00007,
0.00008,
0.00009,
0.00011,
0.00013,
0.00014,
0.00015,
0.00017,
0.00020,
0.00023,
0.00027,
0.00028,
0.00028,
0.00027,
0.00025,
```

```
0.00024,
0.00023,
0.00025,
0.00026,
0.00028,
0.00029,
0.00031,
0.00033,
0.00035,
0.00038,
0.00042,
0.00046,
0.00051,
0.00056,
0.00062,
0.00068,
0.00075,
0.00082,
0.00090,
0.00100,
0.00112,
0.00127,
0.00143,
0.00160,
0.00178,
0.00197,
0.00218,
0.00240,
0.00264,
0.00292,
0.00323,
0.00360,
0.00405,
0.00456,
0.00514,
0.00576,
0.00639,
0.00702,
0.00765,
0.00827,
0.00893,
0.00965,
0.01049,
0.01143,
0.01248,
0.01365,
0.01492,
```

```
0.01628,
0.01776,
0.01938,
0.02115,
0.02312,
0.02532,
0.02782,
0.03071,
0.03419,
0.03849,
0.04380,
0.05021,
0.05770,
0.06595,
0.07460,
0.08343,
0.09236,
0.10162,
0.11151,
0.12234,
0.13427,
0.14748,
0.16158,
0.17659,
0.19251,
0.20934,
0.22707,
0.24568,
0.26516,
0.28547,
0.30658]
```

Zostały utworzone puste listy, do których będa zapisywane wyniki obliczen w każdej iteracji. Z tych list będzie utworzona tabela.

[3]:
```
ax=[]
Ax=[]
kx=[]
pp=[]
```

Pierwsza funkcja służy do wyliczenia czynnika dyskontującego dla podanej stopy procentowej (i) z poniższego wzoru:

$$V = \frac{1}{1+i}$$

Funkcja zwraca wynik działania dla podanego i.

[4]:
```
def CzynnikDyskontujacy(i):
```

```
    return 1/(1+i)
```

Do zmiennej V przypisany zostaje wynik wyżej opisanej funkcji dla i=0.1

[5]:
```
V=CzynnikDyskontujacy(0.1)
```

Kolejna funkcja zwraca wartość prawdopodobieństwa przeżycia k lat, wyliczanego następującym wzorem:

$$ {}_kp_x = \frac{l_{x+k}}{l_x} $$

Funkcja przyjmuje parametry k,n,lx, czyli kolejno: który rok trwania ubezpieczenia,na ile lat dana osoba sie ubezpieczyła i liste ze średnimi liczbami osób dożywających wieku x.

[6]:
```
def kPx(k,n,lx):
    p=[]
    for i in range(0,n-k):
        p.append(lx[30+k+i]/lx[30+k])
    return p
```

Funkcja Ubez wylicza obecną wartość netto ubezpieczenia na życie i dożycie. Ta funckja przyjmuje dwa parametry takie same jak poprzednia oraz dwa nowe czyli px i qx, przy czym px to lista wartości prawdopodobieństw przeżycia kolejnych lat , a qx to prawdopodbieństwo, że osoba nie przezyje okresu x, które brana jest z tablic życia. Obecna wartośc netto ubezpieczenia na życie i dozycie jest wyliczna ze woru:

$$ A_{x:\overline{n|}} = A^1_{x:\overline{n|}} + A_{x:\overline{n|}^1} $$

Wzory na poszczególne elementy tej sumy wyglądają nastepująco:

$$ A^1_{x:\overline{n|}} = \sum_{k=0}^{n-1} v^{k+1} \cdot {}_kp_x \cdot {}_kq_x $$

$$ A_{x:\overline{n|}^1} = v^n \cdot {}_np_x $$

Do zmiennej suma dodana zostaje bieżaca wartość netto ubezpieczenia na życie. Na końcu zostaje ona przemnożona przez wartośc sumy ubezpieczenia oraz oddajemy do niej bieżącą wartość netto ubezpieczenia na dożycie. Po wykonaniu tych wszystkich działań zwrócona zostaje bieżąca wartość netto ubezpieczenia na zycie i dożycie.

[7]:
```
def Ubez(k,n,px,qx):
    suma=0
    for i in range(0,n-k):
        suma +=V**(i+1)*px[i]*qx[30+i]
    return 10000*(suma + V**(n-k)*lx[30+(n-k)+k]/lx[30+k])
```

Funkcja Renta przyjmuje parametry, które zostały opisane w poprzednich funckjach. Wylicza ona wartość renty n-letniej, płatnej na końcu każdego roku. Wzór wygląda nastepująco:

$$ a_{x\overline{n|}} = \sum_{k=1}^{n-1} v^k \cdot {}_kp_x $$

Do zmiennej suma dodana zostaje wartość renty n-letniej, płatej na końcu każdego roku.

[8]:
```
def Renta(k,n,px):
    suma=0
    for i in range(0,n-k):
        suma += V**(i)*px[i]
    return suma
```

Funkcja P oblicz wartość P, która wyrażona jest wzorem:

$$P = \frac{A_{x:\overline{n}|}}{a_{x\overline{n}|}}$$

Wartośc P będzie stała, więc zostanie wyliczona raz, przy teście reszty funkcji i program będzię sie do niej odwoływał w trakcie działania.

[9]:
```
def P(A,a):
    return(A/a)
```

Ostatnia funkcja oblicza wartość rezerwy składki netto. Wiliczana jest ona ze wzoru:

$$ {}_kV_x = A_{x:\overline{n}|} - P \cdot a_{x\overline{n}|}$$

[10]:
```
def rezerwa(A,P,a):
    return A-P*a
```

Tutaj nastepuje test fukncji dla k=0 czyli zerowego roku.

[11]:
```
p=kPx(0,20,lx)
A=Ubez(0,20,p,qx)
a=Renta(0,20,p)
```

Przy okazji testu ozstaje wyliczona również wartośc P.

[12]:
```
P=P(A,a)
```

W pętli wywołujemy nasze funkcje i przypisujemy wyniki obliczen dla każdego roku do list przygotowanych na początku.

[13]:
```
for k in range(0,20):
    p=kPx(k,20,lx)
    A=Ubez(k,20,p,qx)
    a=Renta(k,20,p)
    kv = rezerwa(A,P,a)
    ax.append(round(a,2))
    Ax.append(round(A,2))
    kx.append(round(kv,2))
```

Nastepnie generujemy liste zawierająco lata, bedzie ona naszym indeksem w tabeli.

[14]:
```
k = [ i for i in range(0,20)]
```

Tworzymy obiekt DataFrame uzywając do tego funkcji zawartej w pakiecie pandas, podajac nazwy kolumn.

[15]:
```
df = pd.DataFrame(zip(k,ax,Ax,kx),columns=['k','ax','Ax','kv'])
```

Na końcu usuwamy automatycznie wygenerowany indeks i zastępujemy go kolumną z listą k , przygotowana przed chwilą.

[16]:
```
df=df.set_index('k',drop=True)
df
```

[16]:

k	ax	Ax	kv
0	9.33	1517.37	0.00
1	9.17	1660.51	169.79
2	8.99	1818.38	356.96
3	8.79	1992.52	563.29
4	8.57	2184.58	790.72
5	8.33	2396.39	1041.39
6	8.07	2629.99	1317.71
7	7.78	2887.62	1622.29
8	7.46	3171.71	1957.98
9	7.12	3484.97	2327.94
10	6.73	3830.27	2735.61
11	6.31	4210.99	3184.89
12	5.85	4630.75	3680.08
13	5.34	5093.75	4226.02
14	4.78	5604.56	4828.06
15	4.16	6168.20	5492.07
16	3.48	6790.41	6224.73
17	2.73	7477.41	7033.33
18	1.91	8236.07	7925.91
19	1.00	9074.04	8911.42