# Project 2: Collective risk

Manon Arfib, Tami Famewo, Yann Millet, and Jarek Socha

April 2024

## Contents



## 1 Introduction

In this project, we explored the domain of insurance company dynamics through a simulation-based approach. Our task was to assess the probability of an insurance company maintaining continuous positive capital over a specified period, considering various factors such as claim generation, client enrollment, contract renewal, and initial capitalization.

The scenario we're exploring involves clients generating claims following a Poisson process, where the amount of each claim follows a specific distribution. Additionally, new clients enroll annually according to another Poisson process, with a fixed contract amount payable at the time of enrollment. Renewal of contracts by a proportion of existing clients also plays a significant role in the company's financial stability.

Given these parameters, our goal is to develop a code that can compute the probability of the insurance company maintaining the aforementioned capital over multiple years.

## 2 The project

We chose to break down the work into different functions so it would be more easily readable.

The first function simulates the number of claims that n clients make in a year (*claims_per_client*). As written in the instructions, for each client the number of claims in a year follows a Poisson process with rate $\lambda$.

```
claims_per_client<-function(lambda,n){
  #n:number o clients, lambda:rate of the Poisson process that the number of claims
  #per client follows
  #returns a vector with the number of claims that each client made
  return(rpois(n,lambda*1)) #*1 because we want the number of claims in a year
}
```

Then we wrote the function which gives the amount of k claims. We tried two different methods: the inverse transform and the acceptance-rejection. We chose to incorporate the inverse transform method in our global function because it was the most time-effective, as it is described in part 3.

We integrate the pdf of the distribution that the amount for a claim follows, which gives us the cdf of that distribution: $F(x) = -e^{-(x/147)^{21}} + 1$. Thus, we can compute the quantile function: $u \in (0,1), F^{-1}(u) = 147 \times \sqrt[21]{-ln(-u+1)}$. We then apply the inverse transform algorithm.

```
simulate_inv_trans<-function(k){
  #k: number of claims
  # returns a vector with the value of each claims
  return(147*(-log(-runif(k)+1))^(1/21))
}
```

For the acceptance-rejection method, we used the pdf of an Exponential random variable with $\lambda = 1/147$ as the instrumental pdf $g(x)$. We found that $\frac{f(x)}{g(x)} \leq 21$. We then apply the accpetance-rejection algorithm.

```
f_over_Mg<-function(x){
  return((147/7)*((x/147)^20)*exp(-((x/147)^21)+x/147)/21)
}

simulate_acc_rej<-function(k){
  #k: number of claims
  # returns a vector with the value of each claims
  y<-vector(length=k)
  for (i in 1:k){
    y[i]<-rexp(1,rate=1/147)
    while (runif(1)>f_over_Mg(y[i])){
      y[i]<-rexp(1,rate=1/147)
    }
  }
  return(y)
}
```

Even if we could have incorporated them directly in the global function, we chose to write two more intermediate functions: one which gives the number of clients for the following year, and one which gives the capital at the end of the year.

```
capital_end_year<-function(ni,ci,lambda){
  #ni:number of clients on 1 January, ci: capital on 1 January
  #returns the capital on 31 December after all the claims were made
  total_claim_value<-0
  claims_per_client<-claims_per_client(lambda,ni)
  for (number_of_claims in claims_per_client){
    total_claim_value<-total_claim_value+sum(simulate_inv_trans(number_of_claims))
  }
  return(ci-total_claim_value)
}

new_number_clients<-function(nu,p,n_pre){
  #n_pre: number of clients on 31 December, p: proportion of clients who renew their
  #contract, nu:enrolment rate (per year)
  #returns the number of clients on 1 January of the next year
  return(rbinom(n=1,size=n_pre,prob=p)+rpois(1,nu))
}
```

We then only had to assemble all of our functions into a big one that we called *global*. We treated the first

year separately because our function takes as input the number of clients and capital on 1 January of the first year. As soon as the capital of the company becomes negative (if it does), we stop this round of the simulation.

```
global<-function(c0,n0,a,tl,lambda,nu,p,MC){
  #c0: initial capital, n0: initial number of insured clients, a: insurance's annuity,
  #tl: path length, lambda: claims rate, nu: enrolment rate, p: probability of renewing
  #a contract, MC: number of simulations
  #returns the proportion of paths that remained positive during all the years, the
  #mean and the standard error of those paths
  final_capital<-c() #we will only store the final capital for the paths that always
  #stay positive
  for (i in 1:MC){
    n_clients<-n0
    capital<-capital_end_year(n0,c0,lambda)#capital at the end of year 1
    j<-1 #end of year 1
    while(capital>0 & j<tl){
      j<-j+1
      n_clients<-new_number_clients(nu,p,n_clients)#clients at the beginning of the year
      capital<-capital+a*n_clients #capital after cashing the annuity of the clients
      capital<-capital_end_year(n_clients,capital,lambda)
    }
    if (capital>=0){
      final_capital<-append(final_capital,capital)
    }
  }
  return(c(length(final_capital)/MC,mean(final_capital),sd(final_capital)/sqrt(MC)))
}
```

We didn't encounter any major difficulty. Indeed, breaking the code into different functions helped us not getting lost into what we were coding. It also allowed us to build a global function that is very clearly divided into the different steps that actually happen in an insurance company: update of the number of clients, the clients pay the annuity, the claims are filled and paid by the company.

# 3 Results

In this part, we are going to run a few simulations with different parameters and evaluate the results obtained.

## 3.1 Simulating the amount of a claim

First, we study very briefly the random variable which gives the amount of the claims, by testing our two algorithms (inverse transform and acceptance-rejection).

First, we make sure that both algorithms yield results that indeed follow the cdf $F(x) = -e^{-(x/147)^{21}} + 1$. We use the Kolmogorov-Smirnoff test (with the above cdf declared as the null hypothesis).

```
F<-function(x){
  return(-exp(-(x/147)^21)+1)
}

set.seed(1)

ks.test(simulate_inv_trans(100),F)$p.value
```

```
## [1] 0.6058055
```

```r
ks.test(simulate_inv_trans(1000),F)$p.value
```

```
## [1] 0.4158827
```

```r
ks.test(simulate_inv_trans(10000),F)$p.value
```

```
## [1] 0.4603441
```

```r
ks.test(simulate_acc_rej(100),F)$p.value
```

```
## [1] 0.693768
```

```r
ks.test(simulate_acc_rej(1000),F)$p.value
```

```
## [1] 0.8873634
```

```r
ks.test(simulate_acc_rej(10000),F)$p.value
```

```
## [1] 0.3810858
```

As we can see, no matter the number of data on which the test is carried, the p-value is quite high so we can't reject the null hypothesis. In other words, we checked that both algorithms produce data that indeed follows the distribution we wanted.

We now compared the execution time of both algorithms:

```r
require(microbenchmark)
```

```
## Le chargement a nécessité le package : microbenchmark
```

```
## Warning: le package 'microbenchmark' a été compilé avec la version R 4.3.3
```

```r
set.seed(1)
microbenchmark(simulate_inv_trans(1000),simulate_acc_rej(1000))
```

```
## Unit: microseconds
##                       expr     min       lq      mean   median       uq      max
##  simulate_inv_trans(1000)   165.6   167.65   186.721   175.95    184.2    353.9
##    simulate_acc_rej(1000) 75391.4 94425.05 99162.587 99129.75 101919.1 127837.3
##  neval
##    100
##    100
```

we can very clearly see on the benchmark that simulating with the acceptance-rejection method takes a lot more time. As both algorithms yield results that follow the right distribution, we chose to implement our global function with the inverse transform method.

## 3.2 Global function

In this new version of the report, we increased by a lot the simulation sizes so the results could be way more accurate. We usually use a simulation size of 1000 or 5000 depending on the simulations, as opposed to the previous one which was only 100.

As the expected amount of a claim seems to be around 143, in a year, we expect the insurance company to lose $n_i \times \lambda \times 143$ ($n_i$ being the number of clients at the beginning of the year). At the beginning of the year, the company is expected to earn $(n_{i-1} \times p + \nu) \times a$. This brief analysis will help us make more informed simulations.

We start by simulating a small insurance company with only 15000€ and 100 clients at the beginning of the first year.

```r
set.seed(1)

start_time<- Sys.time()
global(15000, 100, 143, 3, 1, 75, 0.5, 5000)
```

```
## [1]    0.44860 2779.55109   25.09747
```

```r
end_time<-Sys.time()
end_time-start_time
```

```
## Time difference of 3.212951 secs
```

With such parameters, the number of clients increases slightly with the first years. We expect the clients to make 1 claim per year ($\lambda = 1$) of 143€, and the annuity is $a = 143$. Thus, we expect that the assurance company won't lose of earn too much money. However, the first year we expect around 14300€ to be spent by the company which may lead to a non negligible number of negative paths (as the initial capital was only 15000€ after the first annuity was paid). Indeed, 44.86% of the paths remained positive throughout the 3 years of the simulation. The average final capital is 2780€ with a standard error of 25€.

We can check that increasing the value of the initial capital without changing any other parameter (see simulation below) indeed leads to a way higher fraction of paths remaining positive (94% in the following simulation). That is mostly because the risk of the company getting a negative capital in the first year are way lower. The average final capital is also higher (which was expected as this simulation represents a balanced situation between the claims and the annuity).

```r
set.seed(2)

start_time<- Sys.time()
global(20000, 100, 143, 3, 1, 75, 0.5, 5000)
```

```
## [1]    0.97480 5760.60838   36.54719
```

```r
end_time<-Sys.time()
end_time-start_time
```

```
## Time difference of 4.436576 secs
```

We now simulated a way bigger company, with way more clients (50 000 the first year) and a way bigger capital (21.6 million the first year ). We also increased the number of claims per clients to 3. We still wanted to represent a quite balanced situations so we increased the annuity to 430€. We carried the simulation on a time length of 10 years. In order for the number of clients not to change too much, we considered that 75% of them chose to renew their contract and 12 500 joined the company each year.

```r
set.seed(3)

start_time<- Sys.time()
global(21600000, 50000, 430, 10, 3, 12500, 0.75, 1000)
```

```
## [1]      0.711 262108.662   4426.006
```

```r
#we only simulated with 1000 simulations because it takes really long to run
end_time<-Sys.time()
end_time-start_time
```

```
## Time difference of 21.48113 mins
```

This time, the algorithm takes a lot more time to run (more than 26 minutes...). The average final capital is 262 108€ which is slightly higher than the initial margin of the insurance company ($21600000 - 50000 \times 143 \times 3 = 100000$). This shows that this situation corresponds roughly to an equilibrium situation: over the course of 10 years, the insurance company makes a profit of only around 162 000€ so the clients pay a fair price.

Increasing the value of the initial capital would ensure that more paths would be positive the whole time without increasing much the difference between the final capital and the initial margin. So it would be wiser for an insurance company to start with a higher capital.

The standard error of the mean value of the final capital is 4426€, which corresponds to 1.68% of the mean. This isn't too high, but requires for the program to run during a very long time.

We now wanted to explore the cases in which the company either charges a lot or on the contrary doesn't charge enough.

We started with the case of a company which charges a lot more than the cost of the claims. We considered a company of medium size so it wouldn't have to run during too long. We considered a company with an initial capital of 2 500 000€ and 5 000 clients at the beginning of year 1. We still expect 3 claims per clients per year, but this time we raise the annuity to 450€. As the price of the annuity is quite high compared to what the company spends, we can imagine that another company will offer lower prices, so we consider that only 65% of the clients renew their contract and we expect 1500 new clients to join the company each year.

```
set.seed(4)

start_time<- Sys.time()
global(2500000, 5000, 450, 10, 3, 1500, 0.65, 100)
```

```
## [1]       1.00 1145099.09    5201.66
```

```
end_time<-Sys.time()
end_time-start_time
```

```
## Time difference of 13.72912 secs
```

The initial capital was big enough that it was unlikely that the company would get a negative capital during the first year ($25000000 - 5000 \times 3 \times 143 = 355000$€ of margin). For the following years, the high price of the annuity ensures that it is even more unlikely that the capital of the company will get negative. Therefore, just 100 simulated paths (so the program wouldn't run for too), all of them were positive. The average final capital is 1 145 099€, which is way higher than the original margin: the insurance company makes a lot of money on the backs of its customers.

However, the standard error is quite high: 5202€. In order to reduce it, we increase the number of simulated paths to 1000 (simulation below).

```
set.seed(4)

start_time<- Sys.time()
global(2500000, 5000, 450, 10, 3, 1500, 0.65, 1000)
```

```
## [1]       1.000 1154810.203    1647.103
```

```
end_time<-Sys.time()
end_time-start_time
```

```
## Time difference of 2.295755 mins
```

With 1000 simulated paths, the proportion of them always positive is still 100%. The average value of the final capital is roughly the same as the previous simulation, however the standard error is way smaller now (it was divided by more than 3 and corresponds now to only around 0.15% of the average value). However, the execution time was multiplied by roughly 10 (which was to be expected because the number of simulations was multiplied by 10).

The high price of the annuity ensures that the years following the first one won't be often the cause for the insurance getting a negative capital. Therefore, changing the value of the initial capital only will have a huge

impact on the proportion of positive paths. So we then tried to reduce the value of the original capital to 2.16 million.

```
set.seed(4)

start_time<- Sys.time()
global(2160000, 5000, 450, 10, 3, 1500, 0.65, 1000)
```

```
## [1]      0.752 820699.198   1636.672
```

```
end_time<-Sys.time()
end_time-start_time
```

```
## Time difference of 1.799593 mins
```

With such a configuration, 75.2% of the paths remained positive throughout the 10 years of the simulation. However, the average value of the final capital is 820 699€, which is way higher than the expected final capital at the end of year 1 ($2160000 - 5000 \times 3 \times 143 = 15000$). This shows that only the first one is crucial to know whether the capital will remain positive. When charging such a high annuity, the insurance company makes a lot of money if it can go through the first year.

On the contrary, we will now deal with the case of a middle-sized company not charging enough (maybe because the estimations of the number of claims per clients weren't very good for example). We consider the same initial size of the company as the previous one, but this time the annuity is 424€ (which is very close to 429€ which would be roughly the balance price. We can expect that more people will want to stay with this company so we chose 85% of people staying and 2 000 new people joining it each year.

```
set.seed(5)

start_time<- Sys.time()
global(2500000, 5000, 424, 10, 3, 2000, 0.85, 1000)
```

```
## [1]     0.0410 34883.9302   758.7824
```

```
end_time<-Sys.time()
end_time-start_time
```

```
## Time difference of 3.654822 mins
```

Now the results are very different from the previous ones. Only 4.1% of the paths stay positive during 10 years. And the average final capital is 34 884€, which is way smaller than the expected capital at the end of the first year ($2500000 - 5000 \times 3 \times 143 = 355000$). This means that the capital decreases with each year passing.

We can also note that the standard error of the estimation has increased a lot compared to the previous simulation. It is now 759€, which corresponds to 2.17% of the mean remaining capital on the positive paths, as opposed to the previous one which corresponded to only 0.20% of the corresponding mean. This is due to the small number of positive paths.

This shows that choosing the value of the annuity wisely is absolutely necessary. If we run the same simulation with an annuity of 423€ (which is very close to the previous one), only 0.2% (2 paths out of the 1000 simulated) stay positive throughout 10 years (see simulation below).

```
set.seed(6)

start_time<- Sys.time()
global(2500000, 5000, 423, 10, 3, 2000, 0.85, 1000)
```

```
## [1]     0.002 25985.277  1131.007
```

```r
end_time<-Sys.time()
end_time-start_time
```

```
## Time difference of 3.22754 mins
```

# 4 Conclusions

The results detailed above show that there are two crucial points that an insurance company must pay attention to. First, it must make sure that the value of its initial capital is high enough. If it is not the case, the company might go banckrupt in the first year no matter the annuity. And then, the assurance company must set the right annuity price. If it is too low the company will eventually go banckrupt (even if it might be in a long time), but if it is too high, the number of clients renewing their contract or joining the company might decline.

It would have been more realistic to set the number of people joining the insurance company and the number of people renewing their contract as dynamic variables (function of the prices for example).

For the cases where the number of positive paths is low, the standard error is usually quite high. In those cases, it could have been a good idea to implement a variance reduction technique.

# 5 References

[1] assureurpro. 2022. "Classement Des Compagnies d'assurance En France 2022."

[2] Cascos, Ignacio. 2024. "Efficiency Improvement Techniques." Simulation in Probability; Statistics, BSc Applied Mathematics; Computing at UC3M.

[3] Cascos, Ignacio. 2024. "Simulating Random Variables and Vectors." Simulation in Probability; Statistics, BSc Applied Mathematics; Computing at UC3M.