ASC project by Bram Otten (s2330326) [Oct 2020]

Introduction

An insurance company provides professional liability insurance (PLI) and worker's compensation (WC) insurance. These two "business lines" do not seem independent and thus do seem risky. (E.g., the maximum PLI and maximum WC claim correspond to the same client.)

The insurance company wants to be (re)insured, desiring a policy that only pays out when the combined PLI and WC claim of a single client exceeds a threshold $t \in \{100, 110, \ldots, 200\}$. (Conversely and as introduction of some notation: the insurance company will pay the claim itself when $X_1 + X_2 \le t$; X_1 and X_2 denote the amount of the PLI and WC claims in million Euros respectively.)

The reinsurance company asks $P(t) = 40000 \exp(-t/7)$ million Euros for this policy. The (original) insurance company is willing to buy the policy if $P(t) < V(t) = \mathbb{E}(X_1 + X_2)1_{X_1+X_2>t}$. Since the future values of X_1 and X_2 are not known, $V(t)$ must be approximated somehow – simulation will be used in this project. $P(t)$ and the result of *not* using simulation but only real historical data to come up with $V(t)$ are shown in figure 1. $t < 140 \iff V(t) > P(t)$ then (a discrete boundary because of the indicator and lack of randomness).

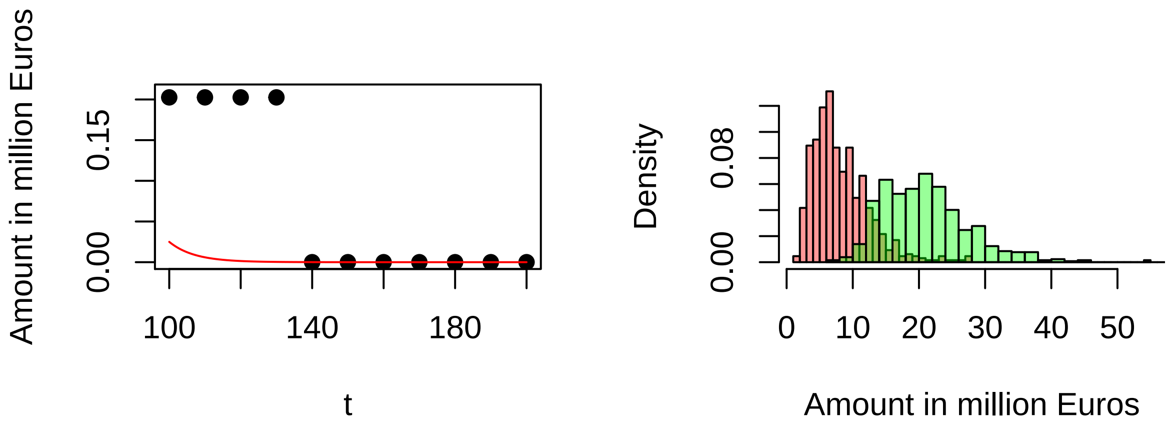


Figure 1: **Left:** Black dots denote values of $V(t)$ based (naively) on only our historical data. Red line $P(t) = 40000 \exp(-t/7)$. $t < 140 \iff V(t) > P(t)$ then (a discrete boundary because of the indicator and lack of randomness). **Right:** red and green histograms of historical realizations of X_1 and X_2 respectively.

This report will describe a modelling process the (original) insurance company may use to model $V(t)$ and use the outcome to decide whether insuring itself with this policy is a good idea. The report will not only contain the final answers but also a bit of the journey – if we are allowed to be meta: it will follow the tasks outlined in the exercise sequentially.

Methodology

To approximate $V(t)$, we need a model for the joint distribution F_{X_1,X_2}. The two random variables (again, denoting the PLI and WC business lines) are presumed to *not* be independent.

With $c(u_1, u_2)$ a "copula density" describing the dependence between the marginals (f_{X_1} and f_{X_2}), the joint density

$$f_{X_1,X_2}(x_1,x_2) = f_{X_1}(x_1)f_{X_2}(x_2)c\big(F_{X_1}(x_1), F_{X_2}(x_2)\big).$$

The function c is the joint density of the probability integral transforms $U_1 = F_{X_1}(X_1)$ and $U_2 = F_{X_2}(X_2)$.

We assume the following based on previous experiments (and because we are told to do so in this assignment):

- $f_{X_1}(\cdot;\mu_1,\sigma_1) \sim \text{Lognormal}(\mu_1,\sigma_1)$, $\mu_1 \in \mathbb{R}, \sigma_1 > 0$;
- $f_{X_2}(\cdot;\mu_2,\sigma_2) \sim \text{Lognormal}(\mu_2,\sigma_2)$, $\mu_2 \in \mathbb{R}, \sigma_2 > 0$; and
- $c(\cdot;\theta) \sim \text{Joe}(\theta), \theta \geq 1$. A little more information on this can be found in the documentation of the `copula` package in R and the Wikipedia page for copulas. (There are two URLs hidden in the previous sentence.)

Estimating marginal densities

For both $j = 1, 2$, μ_j and σ_j can be estimated using maximum likelihood estimation. The notation above is equivalent to $\ln(X_j) \sim \text{Normal}(\mu_j, \sigma_j)$, so we can use the known results for a normal distribution.

```r
ln_X1 <- log(data$PLI)
mu1_mle <- mean(ln_X1)
sigma1_mle <- sd(ln_X1)

ln_X2 <- log(data$WC)
mu2_mle <- mean(ln_X2)
sigma2_mle <- sd(ln_X2)

# Analytic solution:
c(mu1_mle, sigma1_mle, mu2_mle, sigma2_mle)
```

```
## [1] 1.9821374 0.5138609 2.9968907 0.3110814
```

We could alternatively let R's `optim()` minimize the negative log likelihood, but that method makes no fewer assumptions. My instinct would be to use what are already the analytic solutions as starting values, so this really makes no sense. It is implemented below anyway. The specific method does not matter because we start in a (local) optimum.

```r
lognormal_ll <- function(theta, x) sum(log(dlnorm(x, theta[1], theta[2])))
theta1_mle <- optim(c(mu1_mle, sigma1_mle),
                    function(theta) - lognormal_ll(theta, data$PLI))
mu1_mle <- theta1_mle$par[1]
sigma1_mle <- theta1_mle$par[2]

theta2_mle <- optim(c(mu2_mle, sigma2_mle),
                    function(theta) - lognormal_ll(theta, data$WC))
mu2_mle <- theta2_mle$par[1]
sigma2_mle <- theta2_mle$par[2]

# "Optimized" solution:
c(mu1_mle, sigma1_mle, mu2_mle, sigma2_mle)
```

```
## [1] 1.9820806 0.5134259 2.9968962 0.3108721
```

The fit (practically of either but specifically of $\mu_{1,\text{MLE}} = 1.98, \sigma_{2,\text{MLE}} = .513, \mu_{2,\text{MLE}} = 3.00$, and $\sigma_{2,\text{MLE}} = .311$) is shown in figure 2.

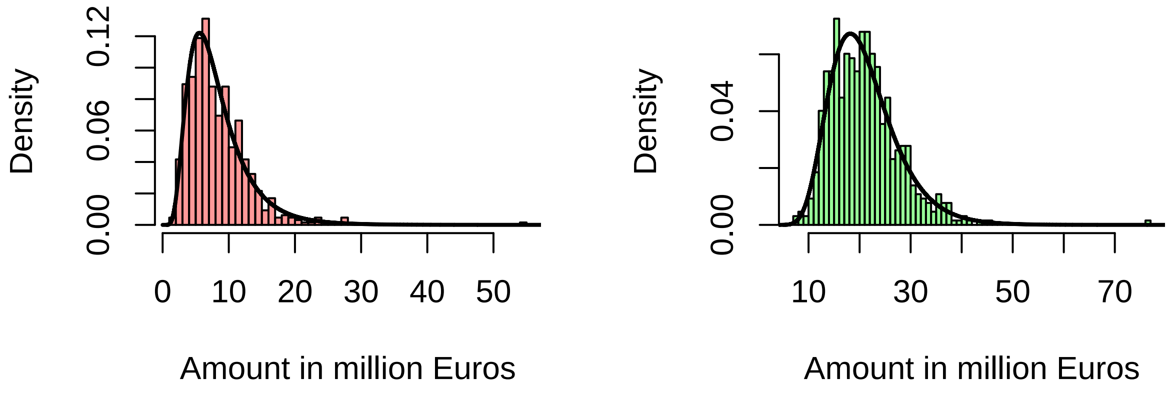


Figure 2: **Left:** PLI / X_1 histogram with MLE lognormal density line. **Right:** WC / X_2 histogram with MLE lognormal density line. The MLE parameters are $\mu_{1,\text{MLE}} = 1.98, \sigma_{2,\text{MLE}} = .513, \mu_{2,\text{MLE}} = 3.00$, and $\sigma_{2,\text{MLE}} = .311$.

Estimating copula density

Remember that:

- The function c is the joint density of the probability integral transforms $U_1 = F_{X_1}(X_1)$ and $U_2 = F_{X_2}(X_2)$; and
- $c(\cdot;\theta) \sim \text{Joe}(\theta), \theta \geq 1$.

We can use pseudo-observations $\hat{U}_j = F_{\hat{\mu}_j,\hat{\sigma}_j}(X_j), j = 1,2$ for a $\hat{\theta}_{\text{MLE}}$ (instead of the actual probability transforms U_j). This cumulative distribution function is the one we have to use in combination with these assumed marginal densities. The historical realizations of X_1 and X_2 are perhaps not the most robust but will result in θ_{MLE}.

We again find the maximally likely parameter by minimizing the negative log likelihood. We know $\theta \geq 1$, and its upper bound is also finite and definitely not > 100 (see figure 5 below – such high values imply practically perfect correlation between X_1 and X_2). R's `optimize()` "is a combination of golden section search and successive parabolic interpolation, and was designed for use with continuous functions", detailed in R.P. Brent's Algorithms for Minimization without Derivatives (2002).

```r
U_hat <- cbind(plnorm(data$PLI, mu1_mle, sigma1_mle),
               plnorm(data$WC, mu2_mle, sigma2_mle))
neg_joe_ll <- function(theta, U = U_hat) {
  - sum(dCopula(U, joeCopula(theta), log = TRUE))
}
theta_opt <- optimize(neg_joe_ll, c(1, 100))
theta_mle <- theta_opt$minimum
theta_mle
```

```
## [1] 1.608187
```

Simulation study

The plan here is to stick to our assumed model but estimate expected values and variances of parameters and payout using different subsets of the data.

Recap of parameter estimation

A function for the complete parameter estimation given some realizations of X_1 and X_2 is:

```r
MLEs <- function(X1, X2) {
  # Note all kinds of assumptions as outlined above.
  # This function is basically a copy-paste of all the above too.
  # It works after clearing the workspace of everything but
  #  "data" and the library.
  ln_X1 <- log(X1)
  mu1_mle <- mean(ln_X1)
  sigma1_mle <- sd(ln_X1)

  ln_X2 <- log(X2)
  mu2_mle <- mean(ln_X2)
  sigma2_mle <- sd(ln_X2)

  lognormal_ll <- function(theta, x) sum(log(dlnorm(x, theta[1], theta[2])))
  theta1_mle <- optim(c(mu1_mle, sigma1_mle),
                      function(theta) - lognormal_ll(theta, X1))
  mu1_mle <- theta1_mle$par[1]
  sigma1_mle <- theta1_mle$par[2]

  theta2_mle <- optim(c(mu2_mle, sigma2_mle),
                      function(theta) - lognormal_ll(theta, X2))
  mu2_mle <- theta2_mle$par[1]
  sigma2_mle <- theta2_mle$par[2]

  U_hat <- cbind(plnorm(X1, mu1_mle, sigma1_mle),
                 plnorm(X2, mu2_mle, sigma2_mle))
  neg_joe_ll <- function(theta, U = U_hat)
    - sum(dCopula(U, joeCopula(theta), log = TRUE))
  theta_opt <- optimize(neg_joe_ll, c(1, 100))

  c(mu1_mle, sigma1_mle, mu2_mle, sigma2_mle, theta_opt$minimum)
}

empirical_MLE_params <- MLEs(data$PLI, data$WC)
empirical_MLE_params
```

```
## [1] 1.9820806 0.5134259 2.9968962 0.3108721 1.6081869
```

Data simulation

We can sample from the joint distribution that follows from a set of parameters as follows:

```r
joint_samples <- function(n, mu1, sigma1, mu2, sigma2, theta) {
  x <- rCopula(n, joeCopula(theta))
  return(cbind(qlnorm(x[,1], mu1, sigma1), qlnorm(x[,2], mu2, sigma2)))
}

set.seed(1)
p <- empirical_MLE_params
samples <- joint_samples(10^4, p[1], p[2], p[3], p[4], p[5])
head(samples)
```

```
##          [,1]     [,2]
## [1,] 3.488920 17.68746
## [2,] 7.537688 15.57849
## [3,] 6.859044 17.46069
## [4,] 9.205684 28.40874
## [5,] 6.361966 22.78426
## [6,] 8.056227 23.59191
```

The distribution of the sum of the sampled values with the MLE parameters unsurprisingly looks a bit like the historical sum of X_1 and X_2 – see figure 3 for a comparison with 10000 samples.

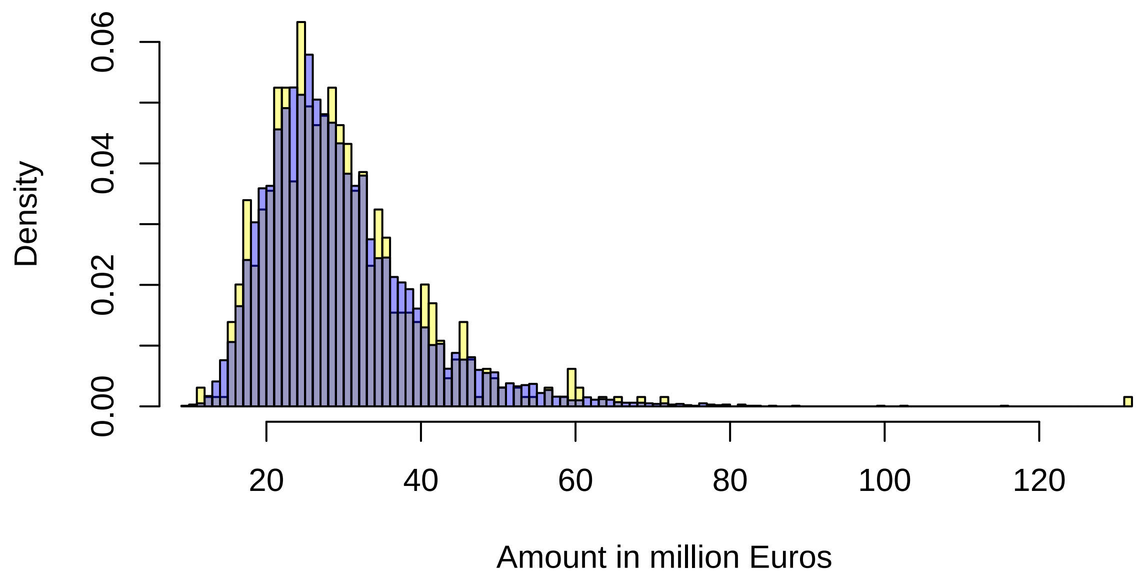


Figure 3: Yellow: `data$PLI + data$WC`. Blue: 10000 estimated $X_1 + X_2$. The parameters are $\mu_{1,\text{MLE}} = 1.98, \sigma_{2,\text{MLE}} = .513, \mu_{2,\text{MLE}} = 3.00$, and $\sigma_{2,\text{MLE}} = .311$.

We see one very large `data$PLI + data$WC` that our sampled values never get close to. To obtain this large value, we can increase the mean(s) or (preferentially) the variance of our X_1 and X_2 compared to the MLE estimates. The effect of this is as expected and as we are familiar with. (Higher means imply higher sums, higher variances imply… higher variances (but uncorrelated ones).) This is illustrated in figure 6.

It is more interesting to report what changing θ does to our sample distribution. Remember that $\theta_{\text{MLE}} \approx 1.61$. Figure 4 shows samples with $\theta = 1.0001$.

Reducing θ seems to decrease the correlation between the random draws. A high X_2 draw is not as likely given a high X_1 draw. A larger $\theta = 10$ predictably does the reverse: it increases correlation between X_1 and X_2 – see figure 5. (Again with 10000 samples – there will always be this many sampled values unless otherwise noted)

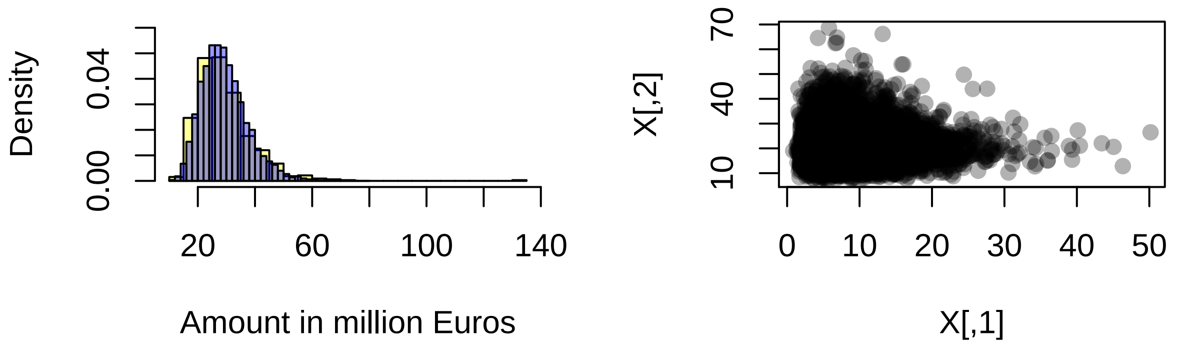


Figure 4: **Left:**Yellow: `data$PLI + data$WC`. Blue: 10000 samples of estimated (almost MLE but with $\theta = 1.0001$) $X_1 + X_2$. **Right:** scatterplot (showing (lack of) correlation) between these sampled X_1 and X_2.

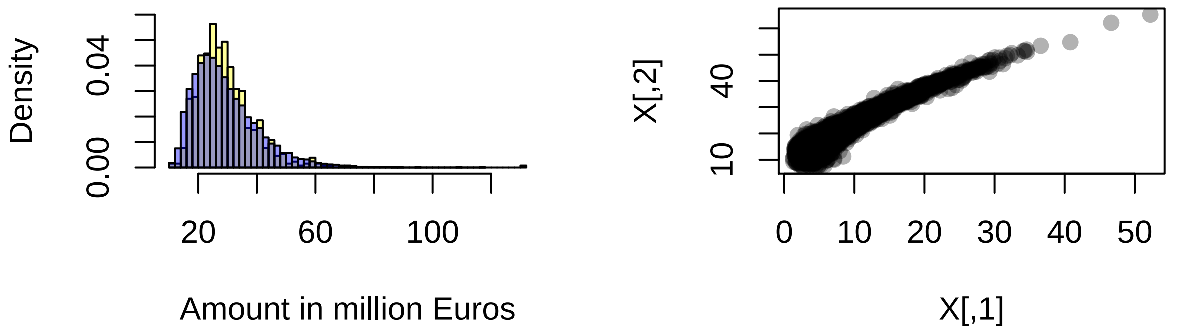
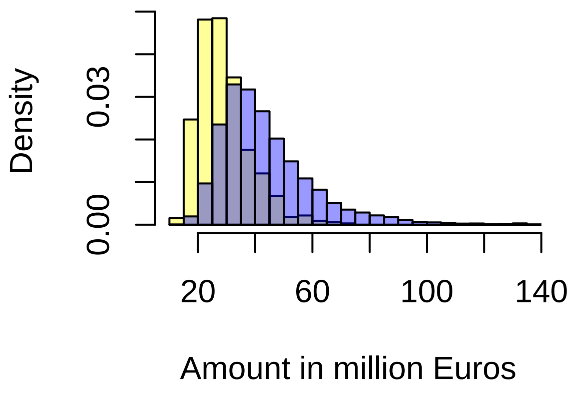


Figure 5: **Left:**Yellow: `data$PLI + data$WC`. Blue: (ml) 10000 samples of estimated (almost MLE but with $\theta = 10$) $X_1 + X_2$. **Right:** scatterplot (showing correlation) between these sampled X_1 and X_2.



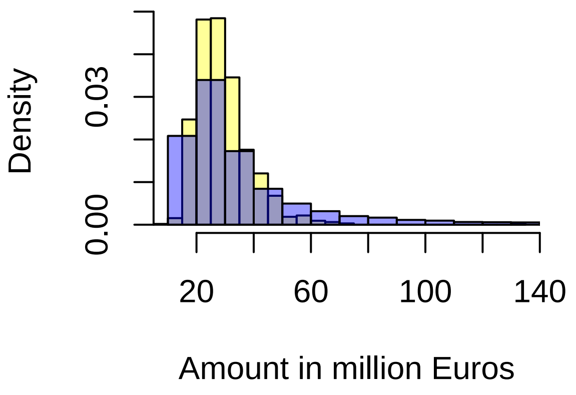


Figure 6: **Left:**Yellow: `data$PLI + data$WC`. Blue: 10000 samples of estimated (almost MLE but with $\mu_1 = \mu_{1,\text{MLE}} + 1$) $X_1 + X_2$. This results in a right-shift. **Right:** Yellow: `data$PLI + data$WC`. Blue: 10000 samples of estimated (almost MLE but with $sigma_1 = \sigma_{1,\text{MLE}} + 1$) $X_1 + X_2$. This results in a fatter (right) tail.

Parameter estimation stability

```
##             mu1 sigma1    mu2 sigma2  theta c. time
## N=0200 0.1497 0.0921 0.0362 0.0252 0.1733  3.4741
## N=0500 0.1051 0.0638 0.0255 0.0151 0.1052  6.9815
## N=1000 0.0607 0.0417 0.0160 0.0104 0.0928 13.5221
```

The root mean square errors of parameter estimation based on differently-sized samples (repeated a hundred times) when compared to the MLE valuesare listed above. We see:

- the time it takes to find parameters does not scale 1-on-1 with the number of samples;
- f_{X_1} seems a bit easier to estimate than f_{X_2};
- all parameters get closer to the value they are in the historical data as the number of samples increases, though this pattern holds by far the least for θ; and
- θ is often relatively far from $\theta_{\text{MLE}} = 1.61$.

Results

Expected payout

In figure 7, we see expected payout $V(t) = \mathbb{E}(X_1 + X_2)1_{X_1+X_2>t}$ for $t = 100, 110, \ldots, 200$ for 10^5 Monte Carlo samples of X_1 and X_2 with model parameters from the observed data.

```
V <- numeric(10)
set.seed(2)
samples <- joint_samples(10^5, p[1], p[2], p[3], p[4], p[5])
S <- samples[, 1] + samples[, 2]
t <- seq(100, 200, length.out = 11)
for (ti in t) {
  V[(ti - 100) / 10] <- mean(ifelse(S > ti, S, 0))
}
plot(seq(100, 200, length.out = 10), V, main = "", xlab = "t",
     ylab = "V(t) and P(t) in million Euros (points and line resp.)",
     ylim = c(0, Pt[1]))
lines(smooth_t, Pt, col = "red")
```

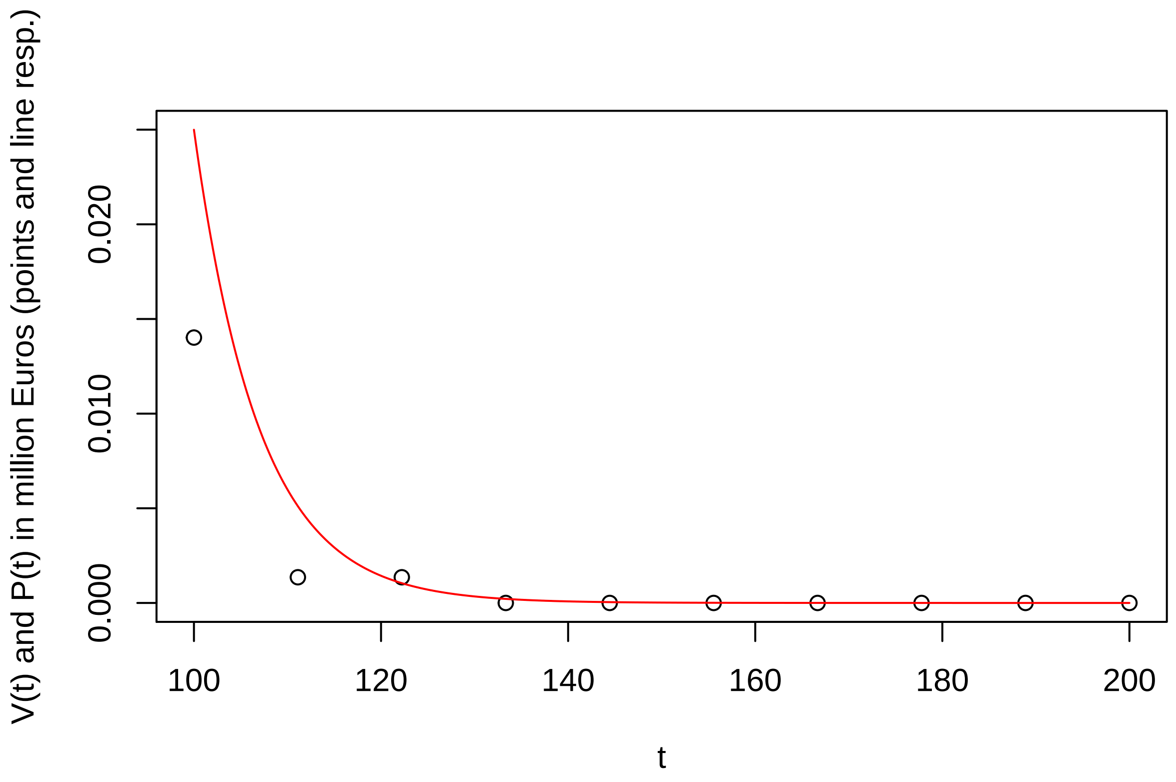


Figure 7: Points represent expected payout $V(t) = \mathbb{E}(X_1+X_2)1_{X_1+X_2>t}$ for $t = 100, 110, \ldots, 200$ for 10^5 Monte Carlo samples of X_1 and X_2 with model parameters from the observed data. Red line $P(t) = 40000 \exp(-t/7)$.

Payouts are improbable – they only occur when $X_1 + X_2$ is "rarely" large. We can use importance sampling to make sure we sample enough of these rarely large values. The actual sampling takes place from a different distribution, but the process is de-biased using the joint density function. This "different distribution" is very similar to our "true" assumed one, but with:

- $\mu_j = \mu_{j,\text{MLE}} + t \div 100$ for $j = 1, 2$ to bias draws in the direction of (variable) t;
- $\sigma_j = \sigma_{j,\text{MLE}} \times 1.5$ for $j = 1, 2$ to make rare draws more likely; and
- $\theta = \theta_{\text{MLE}} \times 1.5$ to increase correlation of X_1 and X_2 draws.

```
joint_density <- function(non_summed_x, mu1, sigma1, mu2, sigma2, theta) {
  dlnorm(non_summed_x[, 1], mu1, sigma1) *
  dlnorm(non_summed_x[, 2], mu2, sigma2) *
  dCopula(cbind(plnorm(non_summed_x[,1], mu1, sigma1),
                plnorm(non_summed_x[,2], mu2, sigma2)),
          joeCopula(theta))
}

importance_V_t <- function(n, mu1, sigma1, mu2, sigma2, theta, t) {
  mu1p <- mu1 + t / 100
  mu2p <- mu2 + t / 100
  sigma1p <- sigma1 * 1.5
  sigma2p <- sigma2 * 1.5
  thetap <- theta * 1.5

  q <- rCopula(n, joeCopula(thetap))
  y <- cbind(qlnorm(q[,1], mu1p, sigma1p), qlnorm(q[,2], mu2p, sigma2p))
  y_sum <- apply(y, 1, sum)

  fx <- joint_density(y, mu1, sigma1, mu2, sigma2, theta)
  fy <- joint_density(y, mu1p, sigma1p, mu2p, sigma2p, thetap)

  return(mean(ifelse(y_sum > t, y_sum, 0) * fx / fy))
}

V <- sapply(t, importance_V_t, n = 10^5,
            mu1 = p[1], sigma1 = p[2], mu2 = p[3], sigma2 = p[4], theta = p[5])
100 + (which(V > P(t)) - 1) * 10 # then buy the insurance policy
```

```
##  [1] 100 110 120 130 140 150 160 170 180 190 200
```

The resulting expected $V(t)$ of 10^5 samples is shown in figure 8. It is smoother than the previous differently obtained $V(t)$ shown in figure 7.

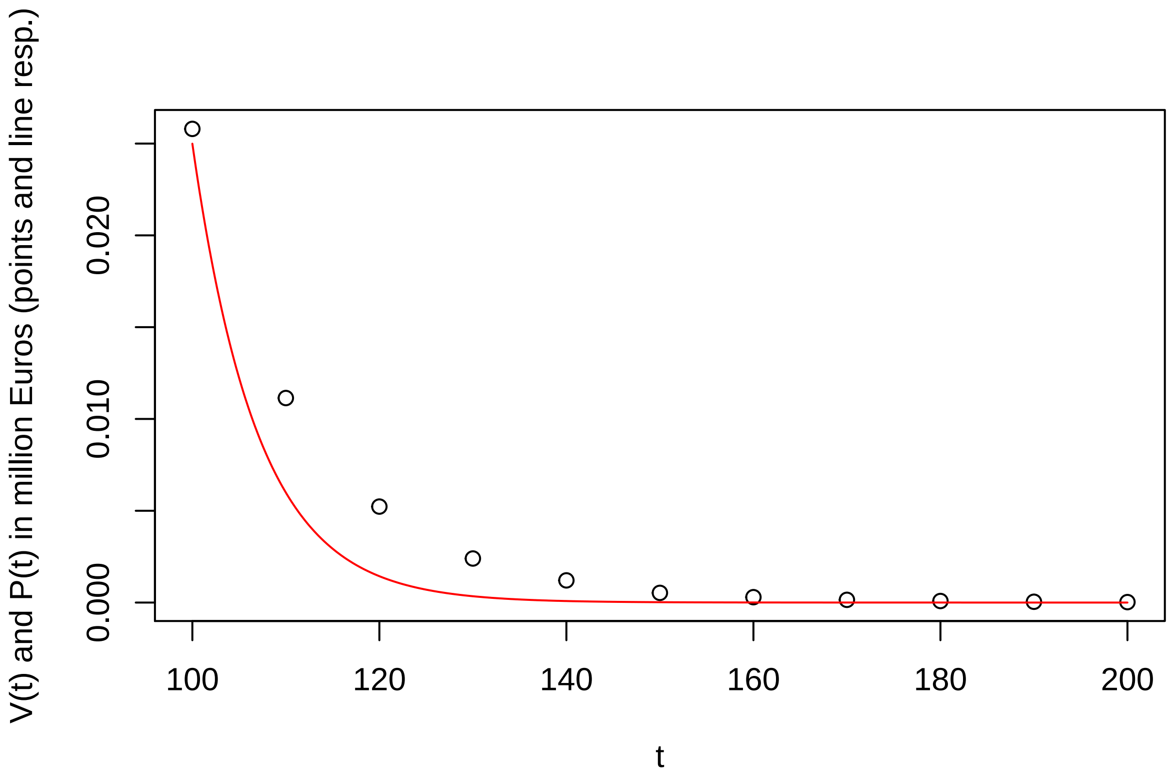


Figure 8: White dots represent expected payout $V(t) = \mathbb{E}(X_1 + X_2)1_{X_1+X_2>t}$ for $t = 100, 110, \ldots, 200$ for 10^5 importance samples of X_1 and X_2 with true parameters ML estimated from the observed data and proposal distribution as described in the text. Red line $P(t) = 40000 \exp(-t/7)$.

Distribution of expected payout

Since our model parameters are estimated, we run our sampling and $V(t)$ estimation multiple times (in this case 100) on a random subset of our data using the bootstrap method to compute (basic) 80% confidence intervals (CIs). This basic 80%-CI is supposed to contain 80% of parameter estimates if we were to estimate many times. For a parameter ϕ and "quantile function for bootstrap replicates" q, it is approximated by $(2\phi - q_{0.1}, 2\phi - q_{0.9})$. An artifact of this generally valid method is that values can become negative. A solution that does not seem necessary in this case is to create a CI for $\log(V)$ and take the exp of the found CI bound. Results are shown in figure 9 and the table below.

```
n <- dim(data)[1]
f <- function() {
  index <- sample.int(n, replace = TRUE)
  p <- MLEs(data$PLI[index], data$WC[index])
  sapply(t, importance_V_t, n = 10^4,
         mu1 = p[1], sigma1 = p[2], mu2 = p[3], sigma2 = p[4], theta = p[5])
}
boot <- t(replicate(fifty * 2, f()))  # 100 rows, length(t) columns
means <- apply(boot, 2, mean)
SDs <- apply(boot, 2, sd)
CIs <- matrix(nrow = length(t), ncol = 2)
for (i in 1:length(t))
  CIs[i, ] <- unname(2 * means[i] - quantile(boot[, i], c(.9, .1)))
table <- cbind(means, SDs, CIs, means - P(t))
colnames(table)[3:5] <- c("80% basic CI low", "... high",
                          "mean-P=E[V(t)]-P(t)")
rownames(table) <- paste(rep("t ="), t)
# V(t) of importance sampling over 100 bootstrap replications:
round(table, 5)
```

```
##             means     SDs 80% basic CI low ... high mean-P=E[V(t)]-P(t)
## t = 100 0.02671 0.00974          0.01497  0.03867             0.00171
## t = 110 0.01174 0.00486          0.00570  0.01737             0.00575
## t = 120 0.00533 0.00224          0.00219  0.00799             0.00390
## t = 130 0.00254 0.00122          0.00096  0.00394             0.00220
## t = 140 0.00123 0.00063          0.00055  0.00191             0.00115
## t = 150 0.00062 0.00037          0.00026  0.00101             0.00060
## t = 160 0.00032 0.00020          0.00011  0.00051             0.00031
## t = 170 0.00017 0.00011          0.00004  0.00028             0.00017
## t = 180 0.00009 0.00006          0.00003  0.00014             0.00009
## t = 190 0.00005 0.00004          0.00001  0.00008             0.00005
## t = 200 0.00003 0.00002          0.00001  0.00005             0.00003
```

These CIs account for estimation error – the error resulting from using the wrong parameters *given* their family – to an extent because they have been obtained with the bootstrap method. If the models do not generalize well to different subsets of the data, the CIs obtained using this method become larger.

The approximation error depends on the distribution families we consider. Since those families have been fixed in the introduction already, we have not reduced it at any point (including now with this bootstrapping).

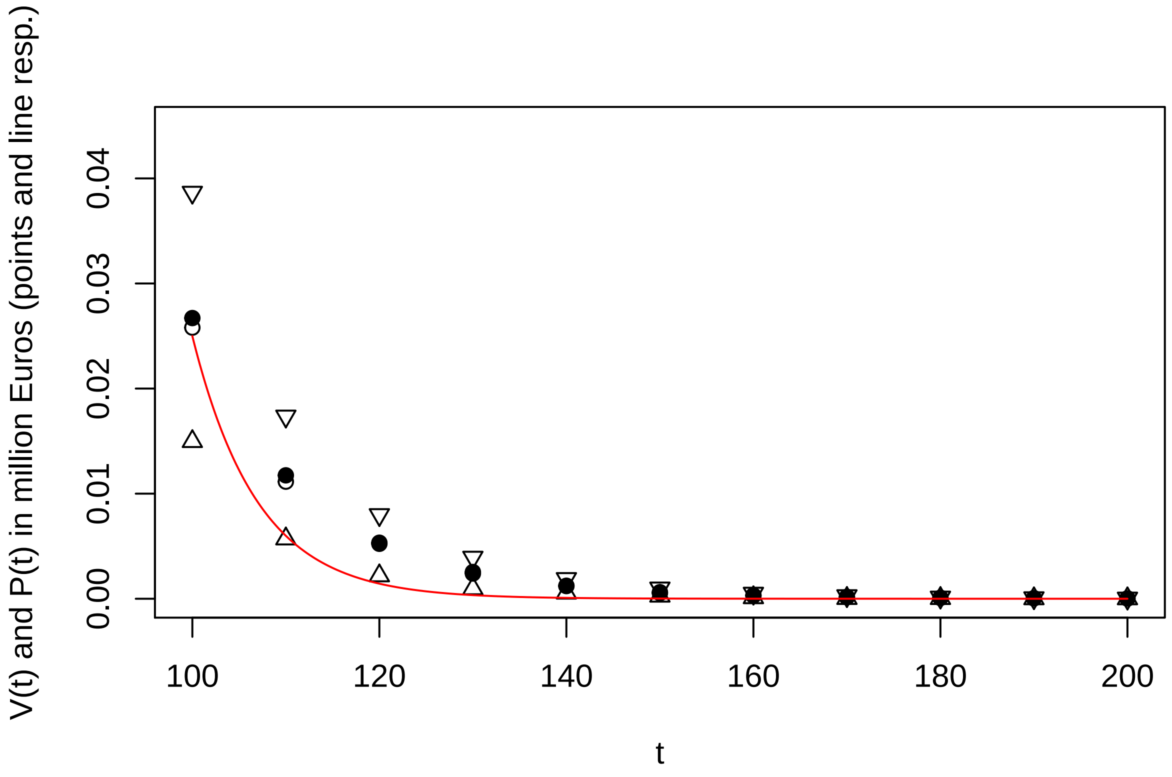


Figure 9: White circles are still expected payout $V(t) = \mathbb{E}(X_1 + X_2)1_{X_1+X_2>t}$ for $t = 100, 110, \ldots, 200$ for 10^4 importance samples of X_1 and X_2 with true parameters ML estimated from the observed data and proposal distribution as described in the text. Black dots are means of 100 bootstrap estimates of $V(t)$ using in principle the same sampling and estimation method. Triangles represent upper and lower bounds of the basic 80% CI. Red line $P(t) = 40000 \exp(-t/7)$.

Final recommendation

The bootstrap method indicates there is only a more than 10% chance of losing a non-trivial amount of money on the insurance policy for a few values of t. (About the trivial amount: e.g. $P(140) = 8.24 \times 10^{-5}$ which corresponds €82.44 (And $\mathbb{E}V(140) - P(140)$ is positive too!))

I would personally only consider *not* taking it at $t = 100$ since the policy is much more expensive there then at larger t. It is still only €25k but at other ts it is practically nothing. To be clear, I would still take the policy even if it had to be at $t = 100$.

This is in accordance with the bootstrap results but can be explained more intuitively too. Even at $t = 100$: if the worst case from the historical data is matched more than once every…

```r
max(data$PLI + data$WC) / P(100)
```

```
## [1] 5251.449
```

… years, the policy will pay off even when *no* other claims in that time exceed t.

The insurance policy (priced at $P(t)$) is simply very cheap compared to the (potential) sizes of the claims the "original" insurance company can face. This is reflected in the expected values we found by simulation too.