End-term project: Advanced Statistical Computing 2020

Yizhen Dai *s2395479*

This document provides an introduction to R Markdown, argues for its benefits, and presents a sample manuscript template intended for an academic audience. I include basic syntax to R Markdown and a minimal working example of how the analysis itself can be conducted within R with the `knitr` package.

Keywords: pandoc, r markdown, knitr

Introduction

The goal

This project aims at solving a modeling problem faced by an insurance company - ANV. Two of ANV business lines, Professional liability insurance (PLI) and Workers' compensation (WC), were affected by a huge claim from one client during the last year. Therefore, ANV comes to a reinsurance company for an insurance policy. To put the problem in a mathematical format, we define the following notation: - X_1: the loss incurred for PLI from ANV clients (in million euros); - X_2: the loss incurred for WC from ANV clients (in million euros); - t: the threshold set by the reinsurance company (in million euros). For some threshold $t = 100, 110, \ldots, 200$, if the total loss incurred for PLI and WC from a certain client exceeds t, ANV pays the claim themselves; Otherwise, the reinsurance company pays the claim; - $V(t)$: the expected over-threshold claims, which is equal to $E[(X_1 + X_2)1(X_1 + X_2 > t)]$; - $P(t)$: the price that reinsurance company asks. $P(t)$ depends on the threshold t as $P(t) = 40000 * e^{-t/7}$.

Our goal is to determine if ANV should buy such policy from the reinsurance company. Of course, the policy is only reasonable if the expected over-threshold claim exceeds the price: $V(t) > P(t)$. The data available is the loss incurred for PLI and WC for all the clients of ANV during last year. With limited data available, we use statistical modeling to approximate $V(t)$ and therefore determine if $V(t) > P(t)$.

Data Explorative analysis

X_1 and X_2 are positively correlated. The correlation between X_1 and X_2 is 0.528. The linear regression line (in red) and the *LOWESS* smoother line (in blue) is shown below.

Models

In order to reflect the dependence in the data, we cannot simulate X_1 and X_2 separately. We use a Copula model to model the joint density f_{X_1,X_2}.

$$f_{X_1,X_2}(x_1, x_2) = f_{X_1}(x_1) f_{X_2}(x_2) c(u_1, u_2)$$

where $u_1 = F_{X_1}(x_1), u2 = F_{X_2}(x_2)$, which are the marginal functions of f_{X_1} and f_{X_2}; $c(u_1, u_2)$ is the joint integral transforms $U_1 = F_{X_1}(X_1)$ and $U_2 = F_{X_2}(X_2)$.

Preliminary experiments suggested the following parametric models:
$f_{X_1}(\cdot; \mu_1, \sigma_1) \sim \text{Lognormal}(\mu_1, \sigma_1), \mu_1 \in \mathbb{R}, \sigma_1 > 0$
$f_{X_2}(\cdot; \mu_2, \sigma_2) \sim \text{Lognormal}(\mu_2, \sigma_2), \mu_2 \in \mathbb{R}, \sigma_2 > 0$

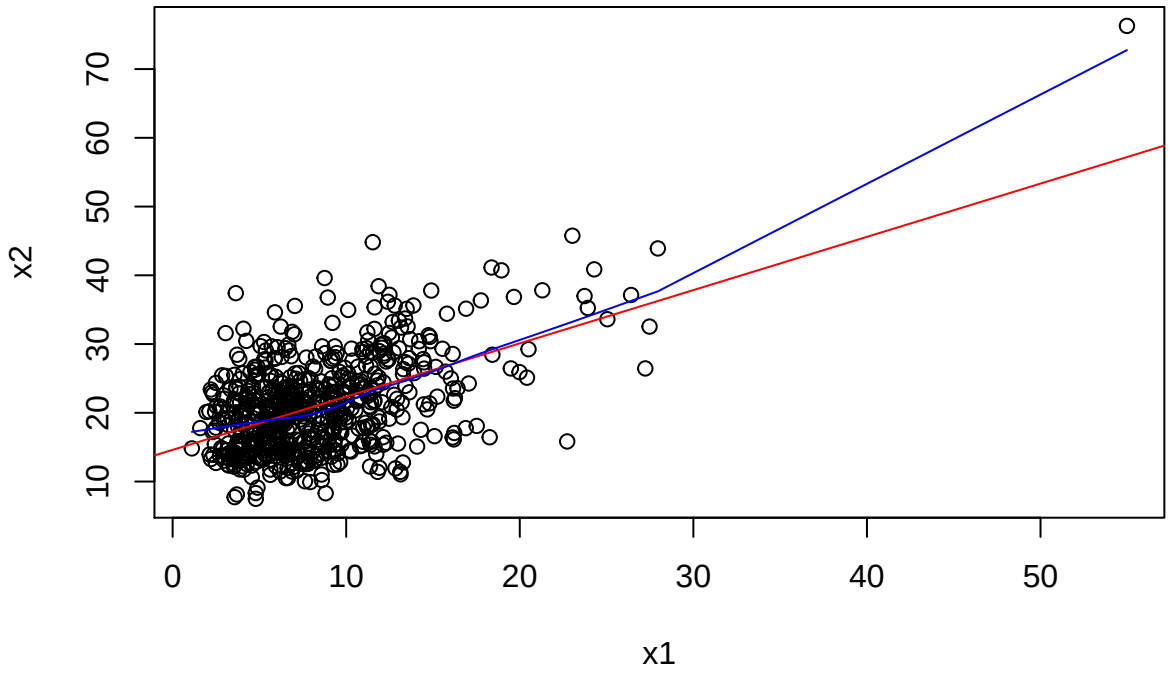


Figure 1: The relationship between X1 and X2

$c(\cdot;\theta) \sim \text{Joe}(\theta), \theta \geq 1$

where a Lognormal distribution is for a random variable whose logarithm is normally distributed; a Joe distribution is one of the most prominent bivariate Archimedean copulas in Copula models.

Methodolgy

Parameter Estimates

Based on the models, we have a total of five parameters to be estimated from the data: $\mu_1, \sigma_1, \mu_2, \sigma_2, \theta$. The maximum likelihood estimation is used to estimate these parameters.

First, we estimate $\mu_1, \sigma_1, \mu_2, \sigma_2$ by finding the values that can maximize the sum of log density of the observed data so the data is most probable under our model assumption f_{X_1}, f_{X_2}. To speed up the optimization, we can use the sample mean and sample standard deviation of the observed data on the log scale.

Afterwards, we estimate θ using the probability integral transforms of the observed data. Such transforms are not observed get pseudo-observations by pluging in the estimated $\mu_1, \sigma_1, \mu_2, \sigma_2$. It might work if our estimated parameters are close to the true parameters.

The codes for parameter estimates are shown below.

```
get_theta <- function(x1, x2, est1, est2) {
    ############################## This function estimate theta using pseudo-observations x1 and
    ############################## observed data est1 and est2 are the estimated parameters of l
    ############################## distribution

    ### get u1 and u2 based on the estimated mu and sigma
    u1 <- plnorm(x1, meanlog = est1[1], sdlog = est1[2])
    u2 <- plnorm(x2, meanlog = est2[1], sdlog = est2[2])
    ### using Joe(theta)-density; evaluated at u; u is a (n x 2) matrix
    g3 <- function(theta) {
        -sum(dCopula(cbind(u1, u2), joeCopula(theta), log = T), na.rm = T)
    }
    ### minimize the negative log likelihood function for the copula function
    nlminb(2, g3, lower = 1 + 10^-8, upper = Inf)$par
}


estimate <- function(x1, x2) {
    ############################## This function estimate the five parameters in our Copula mode
    ############################## are the observed data

    meanlog <- sapply(list(log(x1), log(x2)), mean)
    sdlog <- sapply(list(log(x1), log(x2)), sd)

    ### negative log likelihood function to be minimized
    g1 <- function(beta) {
        -sum(dlnorm(x1, meanlog = beta[1], sdlog = beta[2], log = T), na.rm = T)
    }
```

```
    g2 <- function(beta) {
        -sum(dlnorm(x2, meanlog = beta[1], sdlog = beta[2], log = T), na.rm = T)
    }

    ### Compute estimates for x1 by maximum likelihood method
    est1 <- nlminb(c(meanlog[1], sdlog[1]), g1, lower = c(-Inf, 0), upper = Inf)$par
    names(est1) <- c("mu1", "sigma1")

    ### Compute estimates for x2 by maximum likelihood method
    est2 <- nlminb(c(meanlog[2], sdlog[2]), g2, lower = c(-Inf, 0), upper = Inf)$par
    names(est2) <- c("mu2", "sigma2")

    ### Estimate the parameters of a Joe copula model for (U1,U2).
    est3 <- get_theta(x1, x2, est1, est2)

    return(list(est1 = est1, est2 = est2, est3 = est3))
}
```

The estimated parameters are:

```
est <- estimate(x1,x2)
est
```

```
## $est1
##       mu1    sigma1
## 1.9821375 0.5134642
##
## $est2
##       mu2    sigma2
## 2.9968901 0.3108409
##
## $est3
## [1] 1.608163
```

$\hat{\mu_1} = 1.982, \hat{\sigma_1} = 0.513, \hat{\mu_2} = 2.997, \hat{\sigma_2} = 0.311, \hat{\theta} = 1.608$

Simulated data

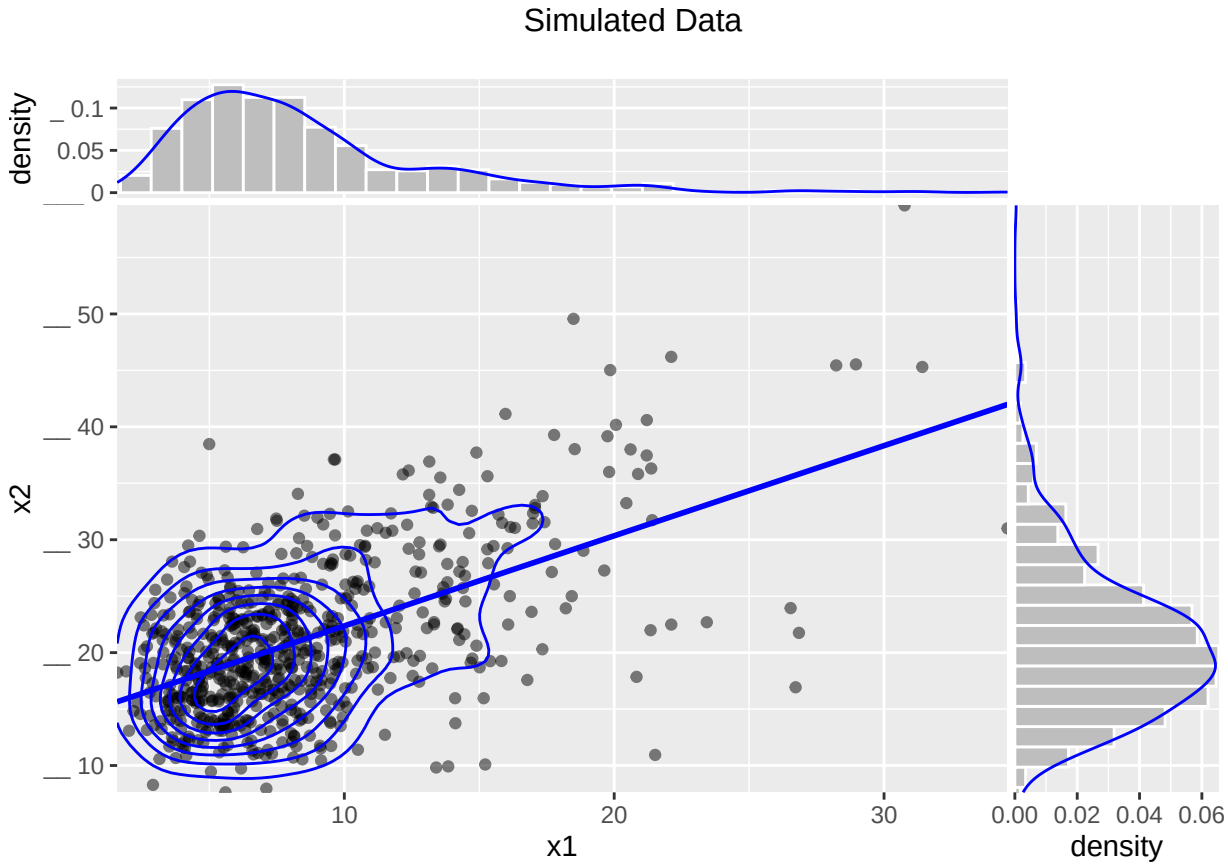
We get the parameter estimates, then we can simulate enough data to get the average cost $V(t)$ over the years.

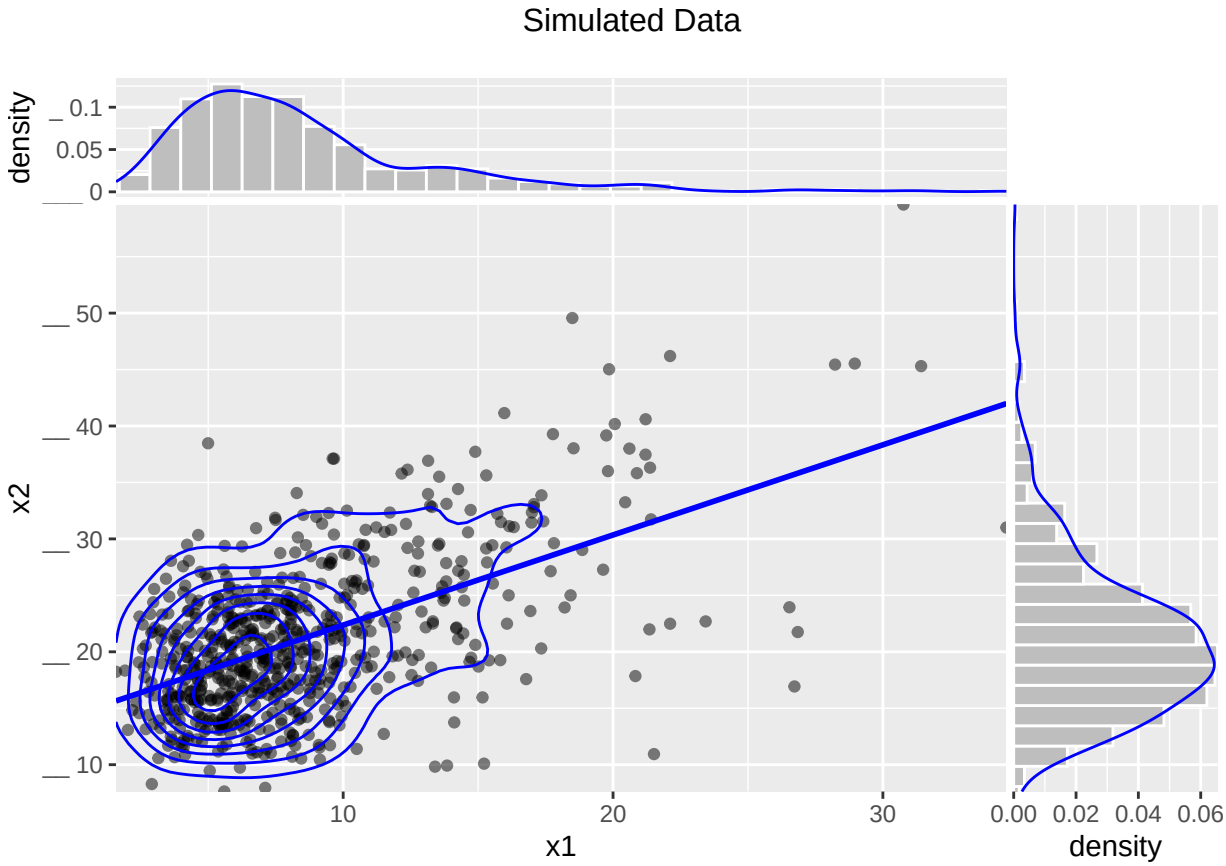
To generate enough data, we need a function that simulates from the joint model for (X_1, X_2) for a given set of parameters $(\mu_1, \sigma_1, \mu_2, \sigma_2, \theta)$. The function is shown below.

```
rmycopula <- function(n, mu1, sd1, mu2, sd2, theta) {
    ############################## This function simulates n samples from a Joe copula with esti
    ############################## parameters

    ### use rCopula(n, joeCopula(theta)) to simulate u1 and u2
```

```
    u <- rCopula(n, joeCopula(theta))  # u is a (n x 2) matrix
    x1 <- qlnorm(u[, 1], mu1, sd1)
    x2 <- qlnorm(u[, 2], mu2, sd2)
    as.data.frame(cbind(x1, x2))
}
```

To evaluate our estimated model, we can compare the simulated data using the above function with the observed data.





We can see the marginal and joint distributions of observed data and simulated data are similar, which is a good sign of our implementation.

We can also check change in data distributions when we tune the parameters using our simulation function. Currently $\hat{\mu_1}, = 1.982, \hat{\sigma_1} = 0.513, \hat{\mu_2} = 2.997, \hat{\sigma_2} = 0.311, \hat{\theta} = 1.608$.

When we increase θ to 4:

We can see with a bigger theta, the data are more correlated with smaller deviation from the linear regression line.

Next, we check the distribution with bigger $\mu_1, \mu_2 : 3, 4$

We can see when we increase μ_1, μ_2, the center of the contour also shifts in the same direction.

Next, we check the distribution with bigger standard deviation of Lognormal distribution: $\sigma_1 = \sigma_2 = 0.8$.

We can see that the data is more dispersed, which is same as expected. We can also observe that a small change in σ_1, σ_2 has a big influence on the extreme values. This is partly because the σ is on the log scale.

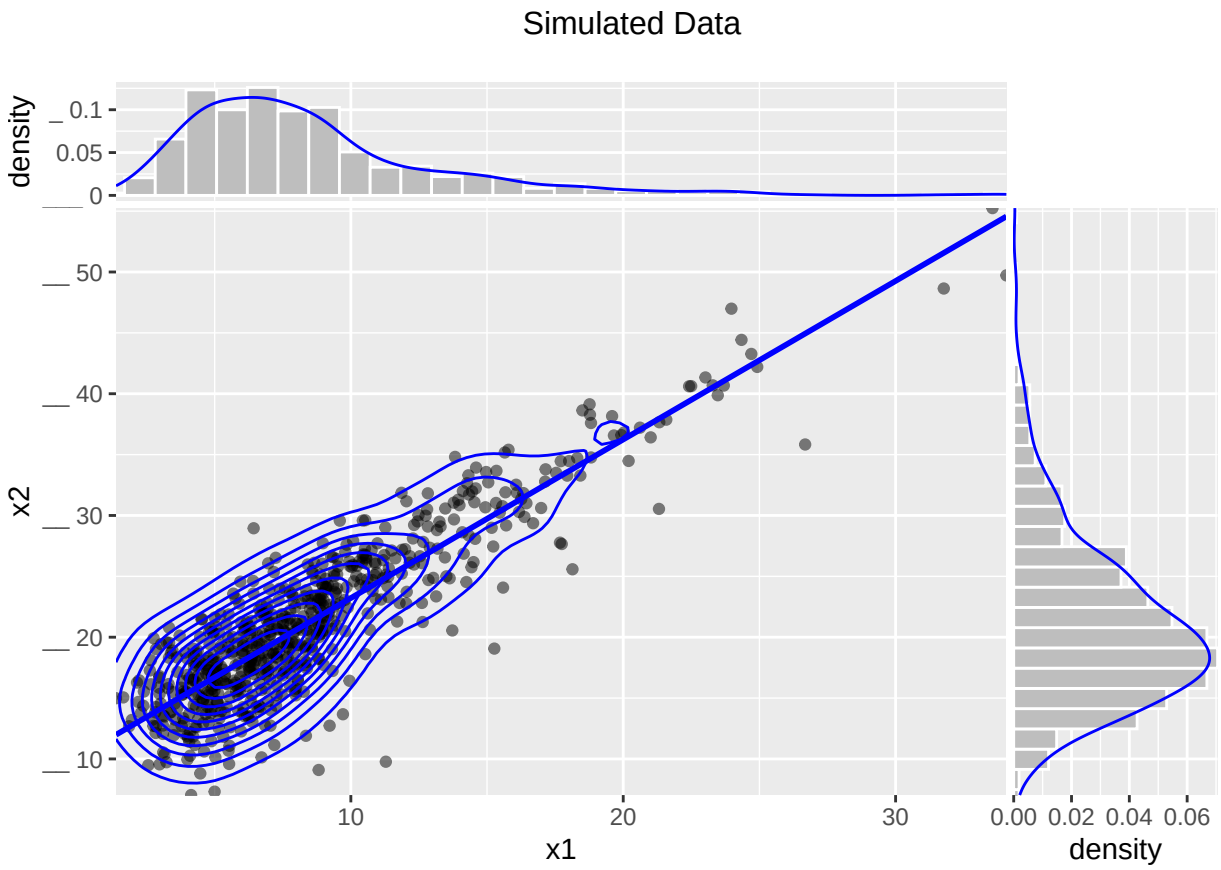


Figure 2: theta = 4

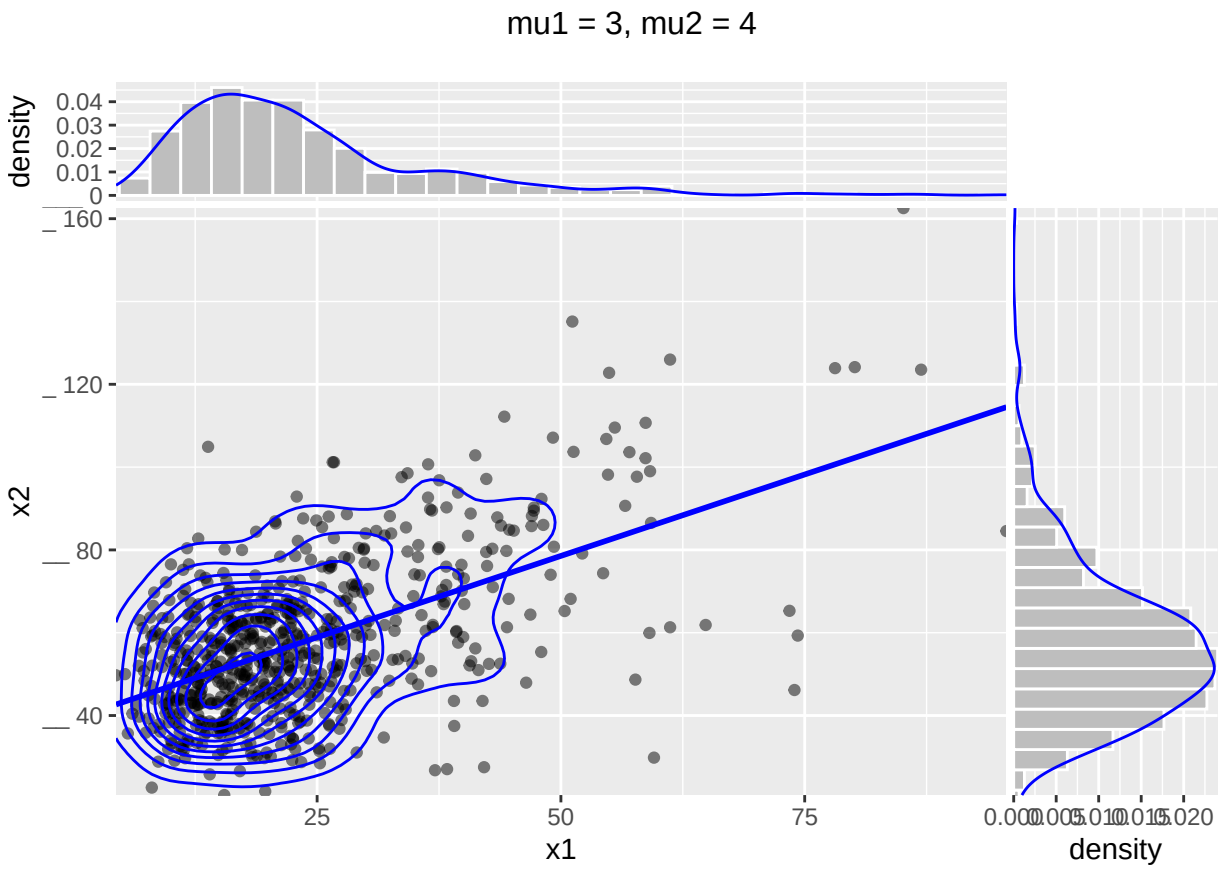


Figure 3: mu1 = 3, mu2 = 4

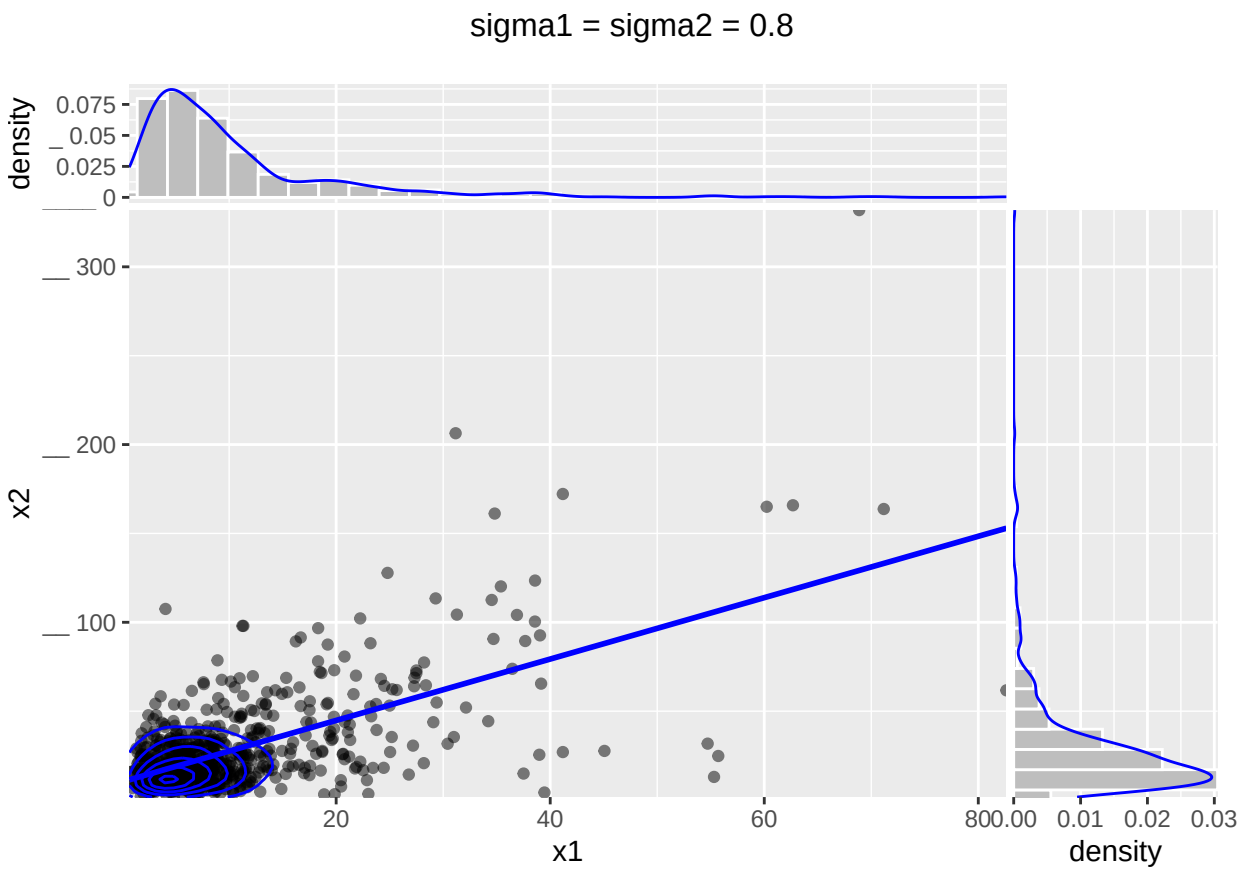


Figure 4: sigma1 = sigma2 = 0.8

Simulation Study

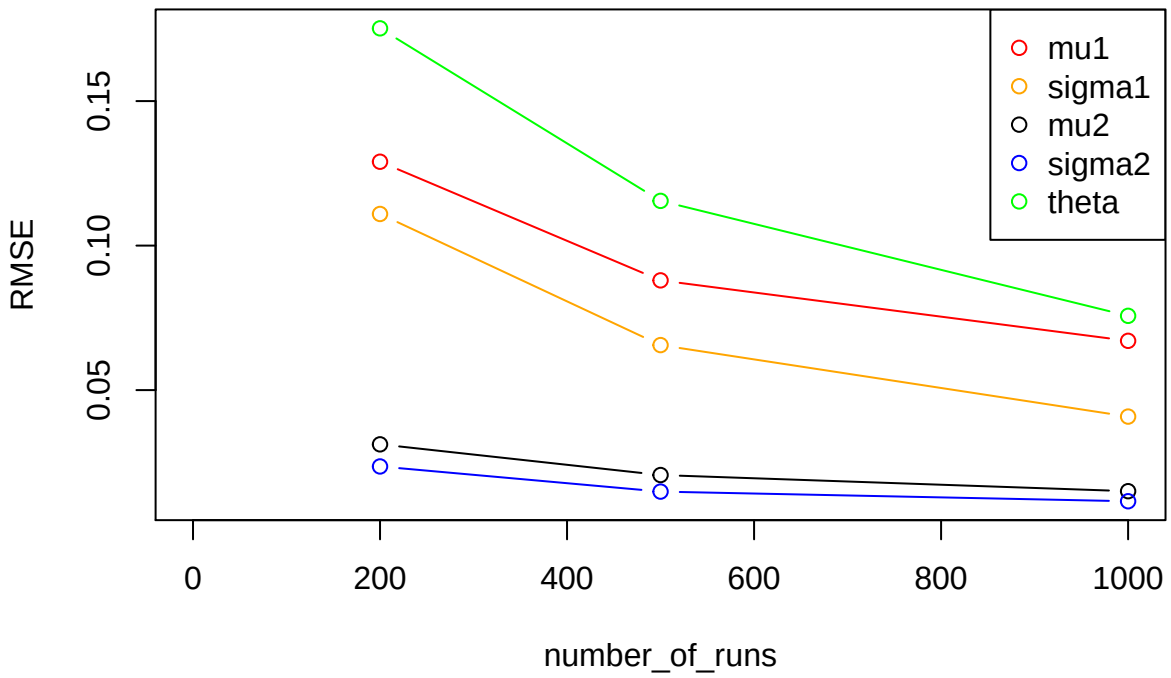
Acuracy evaluation

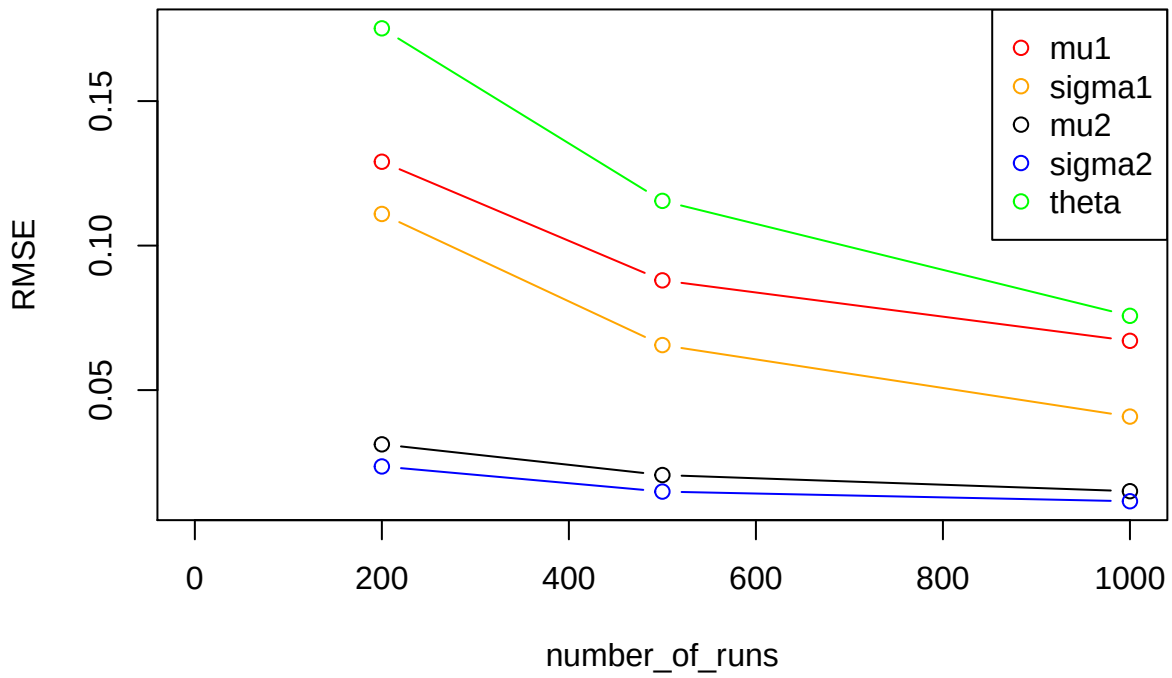
To evaluate how accurate our parameter estimation can be, we conduct a simulation study assuming the following values for true parameters: $\mu_1 = 1, \sigma_1 = 2, \mu_2 = 3, \sigma_2 = 0.5, \theta = 2$.

We compute the RMSE for 100 simulation runs of 200,500 and 1000 data points. The codes for accuracy analysis is shown below:

```
fit_n_times <- function(n){
  ##############################
  # This function estimates the parameters for 100 times and returns the RMSE
  # n: the number of data points in each simulation run
  ##############################

  ### simulate
  sim_data <- lapply(1:100,function(x) rmycopula(n, 1, 2, 3, 0.5, 2))
  results <- c()
  for (i in 1:100){
    results <- cbind(results,
                     unlist(estimate(sim_data[[i]]$x1, sim_data[[i]]$x2)))
  }
  ### calculate RMSE
  (results - c(1, 2, 3, 0.5, 2))^2 %>%   # estimated minus true values
    rowMeans(.) %>%
    sqrt(.)
}

### initiate time and RMSE
time <- numeric(3)
RMSE <- matrix(0, 3, 5)

### conduct simulation for n = 200,500 and 1000
i <- 1
for (n in c(2,5,10) * 100) {
  time[i] <- system.time(RMSE[i,] <- fit_n_times(n))/100 # average computing time
  i <- i + 1
}

### update format for plotting
time <- data.frame(number_of_runs=c(2,5,10) * 100, time=time)
```

The RMSE and average computing time are shown in the following figure.


RMSE
0.05
0.10
0.15
0
200
400
600
800
1000
number_of_runs
mu1
sigma1
mu2
sigma2
theta



We can see that required computation time increase linearly with the number of runs.

RMSE gets smaller with more simulation runs. θ is the hardest parameter to estimate with largest RMSE. μ_1, σ_1 have higher RMSE than μ_2, σ_2 separately. This is probably due to a lower standard deviation for f_{X_2} (0.5 compared to 2 on the log scale). The distribution for f_{X_1} is more dispersed, leading to a larger RMSE for μ_1, σ_1.

Working on the real data: V(t)

Having a feeling of the simulation models, we fit all the five parameters to the observed data. We compute the expected payout of the reinsurance $V(t)$ using parametric Monte Carlo simulation (based on the estimated parameters). We use 10^5 Monte Carlo samples and calculate expected $V(t), t = 100, 110, \ldots, 200$, as the average over the simulation runs.The codes are shown below

```
B <- 10^5  # number of simulation runs

### simulated B data points to save time
simulated_data <- rmycopula(B, est$est1[1], est$est1[2], est$est2[1], est$est2[2],
    est$est3)
total_claim <- rowSums(simulated_data)  # x1 + x2

### replace claims <= t with zeros
V <- sapply(seq(100, 200, 10), function(t) c(total_claim[total_claim > t], rep(0,
    sum(total_claim <= t))))
```

```
Vt <- colMeans(V)  # mean for each client
Vt_sd <- apply(V, 2, sd)  # sd
```

The table and the plot of $V(t)$ for $t = 100, 110, \ldots, 200$ based on 10^5 Monte Carlo samples are shown below.

```
##
## % Table created by stargazer v.5.2.2 by Marek Hlavac, Harvard University. E-mail: hlavac at f
## % Date and time: Thu, Oct 15, 2020 - 02:34:26
## \begin{table}[!htbp] \centering
##   \caption{P(t) and V(t)}
##   \label{}
## \begin{tabular}{@{\extracolsep{5pt}}lccccccc}
## \\[-1.8ex]\hline
## \hline \\[-1.8ex]
## Statistic & \multicolumn{1}{c}{N} & \multicolumn{1}{c}{Mean} & \multicolumn{1}{c}{St. Dev.} &
## \hline \\[-1.8ex]
## P(t) & 11 & 0.003 & 0.008 & 0.00000 & 0.00000 & 0.001 & 0.025 \\
## V(t) & 11 & 0.004 & 0.009 & 0.000 & 0.000 & 0.004 & 0.031 \\
## Std for V(t) & 11 & 0.357 & 0.646 & 0.000 & 0.000 & 0.479 & 1.832 \\
## P(t) \textless  V(t) & 11 & 0.273 & 0.467 & 0 & 0 & 0.5 & 1 \\
## \hline \\[-1.8ex]
## \end{tabular}
## \end{table}
```

Based on the results, buying the reinsurance policy seems a good idea when $t \leq 120$ since the policy price ($P(t)$)is smaller than the potential cost of not buying the policy ($V(t)$). However, the standard deviation is high for $V(t)$. This is because the claim exceeding t is a highly improbable event.

Importance sampling

To handle the highly improbable events, we use importance sampling. The codes are shown below.

```
### new parameters est2 for importance sampling
est2 <- est
### setting higher mu's to make the events more probable
est2$est1[1] <- est2$est1[1] + 0.5
est2$est2[1] <- est2$est2[1] + 0.5


dmycopula <- function(Y, est) {
    ############################## This function calculate the density of the copula model at Y
    ############################## observed data with X1 and X2
    x1 <- Y[, 1]
    x2 <- Y[, 2]
    u1 <- plnorm(x1, meanlog = est$est1[1], sdlog = est$est1[2])
```

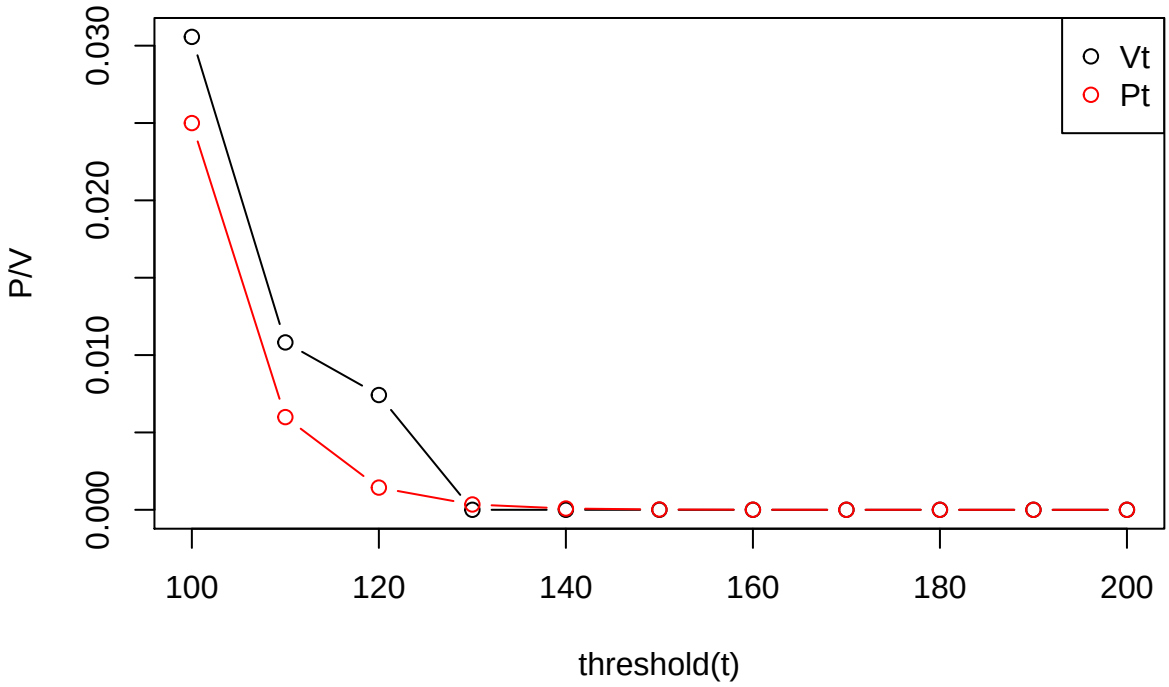


Figure 5: P(t) vs t

```
    u2 <- plnorm(x2, meanlog = est$est2[1], sdlog = est$est2[2])
    dlnorm(x1, est$est1[1], est$est1[2]) * dlnorm(x2, est$est2[1], est$est2[2]) *
        dCopula(cbind(u1, u2), joeCopula(est$est3))
}

simulated_data2 <- rmycopula(B, est2$est1[1], est2$est1[2], est2$est2[1], est2$est2[2],
    est2$est3)
total_claim2 <- rowSums(simulated_data2)

V2 <- sapply(seq(100, 200, 10), function(t) {
    ind <- (total_claim2 > t)  # index for the data with total claim > t
    temp <- simulated_data2[ind, ]  # get Y for dmycopula
    ### importance sampling
    c(total_claim2[ind] * dmycopula(temp, est)/dmycopula(temp, est2), rep(0,
        B - sum(ind)))  # replace claims <= t with zeros
})

Vt2 <- colMeans(V2)  # mean for each client
Vt_sd2 <- apply(V2, 2, sd)  # sd
```

The table and the plot of $V(t)$ for $t = 100, 110, \ldots, 200$ based on 10^5 Monte Carlo samples and importance sampling are shown below.

```
##
## % Table created by stargazer v.5.2.2 by Marek Hlavac, Harvard University. E-mail: hlavac at f
## % Date and time: Thu, Oct 15, 2020 - 02:34:26
## \begin{table}[!htbp] \centering
##   \caption{P(t) and V(t) based on importance sampling}
##   \label{}
## \begin{tabular}{@{\extracolsep{5pt}}lccccccc}
## \\[-1.8ex]\hline
## \hline \\[-1.8ex]
## Statistic & \multicolumn{1}{c}{N} & \multicolumn{1}{c}{Mean} & \multicolumn{1}{c}{St. Dev.} &
## \hline \\[-1.8ex]
## P(t) & 11 & 0.003 & 0.008 & 0.00000 & 0.00000 & 0.001 & 0.025 \\
## V(t) & 11 & 0.005 & 0.009 & 0.00002 & 0.0001 & 0.004 & 0.028 \\
## Std of V(t) & 11 & 0.125 & 0.159 & 0.007 & 0.015 & 0.175 & 0.486 \\
## P(t) \textless  V(t) & 11 & 1.000 & 0.000 & 1 & 1 & 1 & 1 \\
## \hline \\[-1.8ex]
## \end{tabular}
## \end{table}
```

The $V(t)$ based on importance sampling have values when $t > 120$. Based on the results, the company should buy the reinsurance policy regardless of the value of t.

Confidence Intervals

Since we are not sure how confident we are about our estimates, we can to conduct bootstrapping to get confident intervals of $V(t)$. The codes doing a bootstrap method to compute 80% confidence

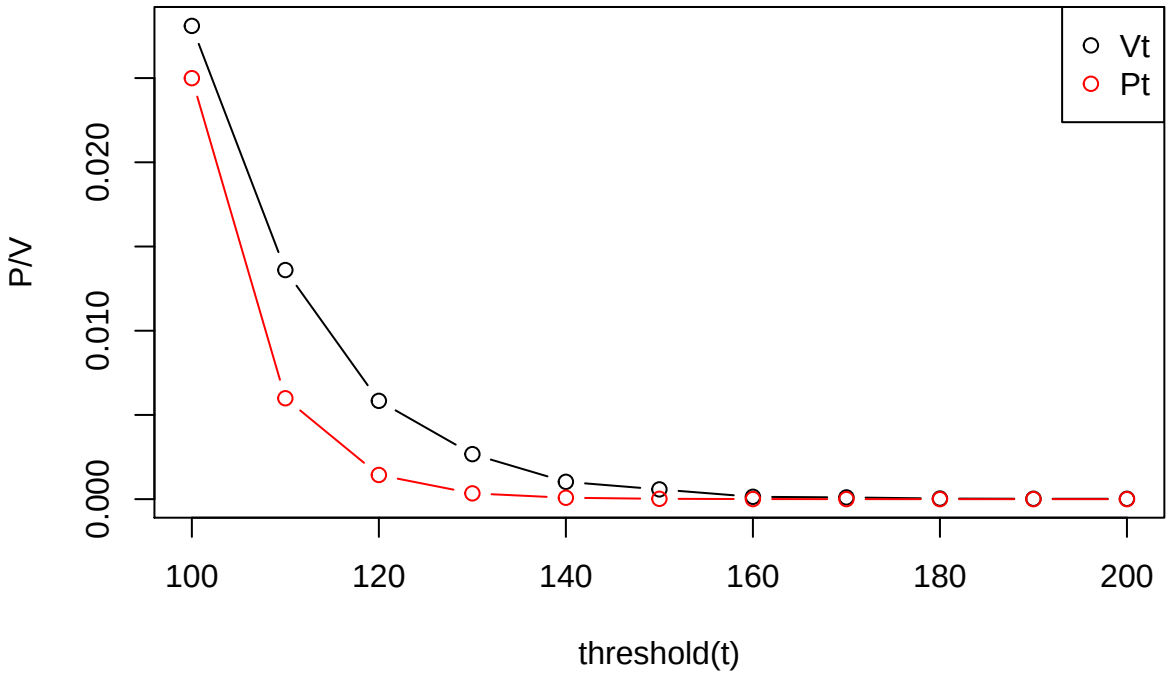
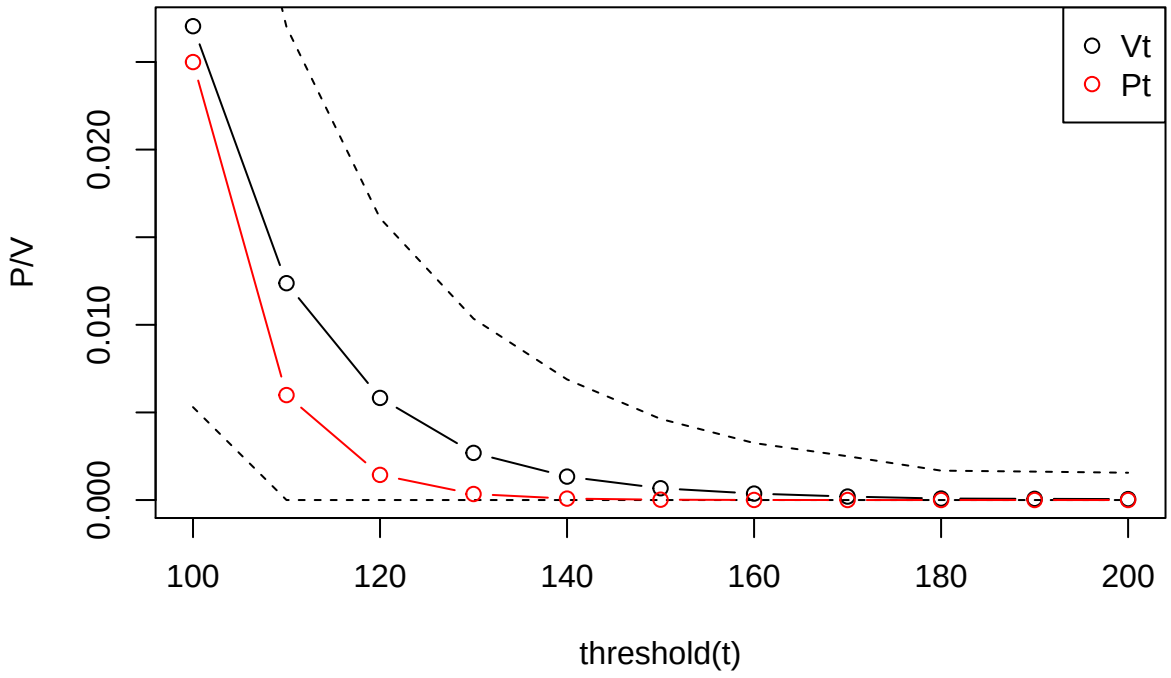


Figure 6: P(t) and V(t) vs t based on importance sampling

intervals for $V(t)$ are shown below.

```
### construct the confident intervals
mean_boot <- apply(V3, 2, mean)
se_boot <- apply(V3, 2, sd)  # Bootstrap se's of the coefficients
### calculate 80% confidence intervals for V(t)
CI_low <- mean_boot + se_boot * qnorm(0.1)
CI_low[CI_low < 0] <- 0
CI_up <- mean_boot + se_boot * qnorm(0.9)
```



Results As you can see from the graph, there is huge uncertainty in the values of $P(t)$. We cannot draw a conclusion that $V(t)$ will be smaller than $P(t)$.

The implementation does account for MC approximation error since we are doing bootstrapping. The implementation does not account for estimation error since we are relying on the existing data (648 data points).