ACTL3162 General Insurance Techniques Assignment

Johnny Wong, z5016960

October 8, 2017

Contents

1 **Task 1** **2**
- 1.1 Summary . . . 2
- 1.2 General procedure . . . 2
- 1.3 Pareto distribution . . . 3
- 1.4 Log-Normal distribution . . . 4
- 1.5 Gamma distribution . . . 4
- 1.6 Calculations for visual analysis . . . 4
  - 1.6.1 P-P plots . . . 7
  - 1.6.2 Q-Q plots . . . 8
- 1.7 Hypothesis tests . . . 8
  - 1.7.1 Pearson's Chi-squared test . . . 8
  - 1.7.2 Kolmogorov-Smirnoff test . . . 10

2 **Task 2** **11**
- 2.1 a) . . . 11
- 2.2 b) . . . 11
- 2.3 c) . . . 12
- 2.4 d) . . . 12
- 2.5 e) . . . 13
  - 2.5.1 (A) . . . 13
  - 2.5.2 (B) . . . 13
  - 2.5.3 Compare and comment on differences in your results relative to part b). . . . 14

3 **Task 3** **15**
- 3.1 a) . . . 15
  - 3.1.1 Distribution of Y_5 . . . 15
  - 3.1.2 Distribution function at $x = 10, 20, \ldots, 50$ . . . 15
- 3.2 b) . . . 15
  - 3.2.1 Discretise S_i and use Panjer's recursion to approximate Y_5 . . . 15
  - 3.2.2 Methodology and assumptions . . . 16
- 3.3 Comparison . . . 17

A **Task 1** **18**
- A.1 MLE for Pareto . . . 18

B **R code** **18**
- B.1 Data pre processing . . . 18
- B.2 Task 1 . . . 19
- B.3 Task 2 . . . 26
- B.4 Task 3 . . . 29

1 Task 1

1.1 Summary

The Maximum Likelihood Estimator (MLE) was found for 3 distributions: Pareto, Log-normal, and Gamma. It was found that the Log Normal distribution fit this data the best.

The following table summarises the estimated parameters and their associated Aikaike Information Criteria (AIC). The AIC captures information concerning the of goodness of fit of the data to the model and the need to build a model with many variables. Since we're fitting all models to the same data, and all three models have 2 parameters to fit, the AIC, BIC, and likelihood measures are all essentially equivalent.

Distribution	Parameter 1	Parameter 2	AIC
Pareto	$\alpha = 51.18$	$\lambda = 200,000$	22,033.600
Log-Normal	$\mu = 8.432$	$\sigma = 0.396$	21,146.657
Gamma	$\alpha = 6.555$	$\beta = 0.001$	21,156.529

Furthermore, the empirical and theoretical distributions are plotted. Through visual means, it is obvious that the Pareto distribution is not an appropriate fit for the data. Additionally, the Log Normal distribution seems to deviate less from the empirical data than the Gamma distribution, suggesting a better fit to the Log Normal distribution.
P-P and Q-Q plots were created and revealed that the Log Normal distribution tends to overestimate the tail, whereas the Gamma distribution underestimates it. In the context of modelling insurance claims, it is better to have a model that overestimates claim size rather than one that underestimates. This leads to higher reserves and a lower probability of ruin.
Hypothesis tests were also conducted, concluding that the data is not Pareto distributed, but could be Log Normal or Gamma. The hypothesis tests favoured the Log Normal distribution over the Gamma. Through the combination of both statistical and visual analysis, it is concluded that the Log-Normal distribution is the most appropriate distribution to fit to the data, followed closely by the Gamma distribution.

1.2 General procedure

The MLE is calculated by first finding the likelihood function. This is simply the product of the probability density function (PDF) evaluated at each data point. Furthermore, each the contribution of the PDF from each datapoint is adjusted for the claim's level of excess.
Noting that every policy has a minimum \$700 excess and that each policy can have an additional excess, if we let the following denote the characteristics of the ith claim,

$$\begin{aligned} \text{Additional excess} &= e_i \Rightarrow \text{Overall excess} = d_i = 700 + e_i \\ \text{Paid amount} &= p_i \\ \text{Claim amount} &= x_i = d_i + p_i \end{aligned}$$

then the likelihood is calculated as,

$$L = \prod_{i=1}^{n} \frac{f(x_i)}{1 - F(d_i)}$$

Where:
$n = 1185$ is the number of claims recorded
f is the PDF
F is the Cumulative Distribution Function (CDF)

The denominator adjusts for the fact that since an observation is not truncated out, it must have a claim size above the deductible.

For computational efficiency, the natural logarithm is applied to the likelihood, resulting in the log-likelihood.

$$l = \sum_{i=1}^{n} [\ln(f(x_i)) - \ln(1 - F(d_i))] \tag{1}$$

If we let $\boldsymbol{\theta}$ represent the vector of distribution parameters, then the MLE, $\hat{\boldsymbol{\theta}}^{MLE}$, is the solution to:

$$\frac{\partial l}{\partial \boldsymbol{\theta}} = \mathbf{0}$$

Where $\mathbf{0}$ is the zero vector.

The AIC is then calculated as:

$$\begin{aligned} \text{AIC} &= -2l + 2k \\ &= -2l + 4 \\ \text{Where} & \\ l &= \text{The log likelihood} \\ k &= \text{The number of parameters of the model} = 2 \end{aligned}$$

1.3 Pareto distribution

The Pareto distribution has two parameters, a shape parameter α, and a scale parameter λ. With these two parameters, the PDF and CDF are

$$f(x) = \frac{\alpha\lambda^{\alpha}}{(\lambda + x)^{\alpha+1}}, \quad x > 0$$
$$F(x) = 1 - \left(\frac{\lambda}{\lambda + x}\right)^{\alpha}, \quad x > 0$$

This yields a log likelihood of

$$l = \sum_{i=1}^{n} [\ln(\alpha) - (\alpha + 1)\ln(\lambda + x_i) + \alpha \ln(\lambda + d_i)]$$

By setting the first derivatives to 0, we can express the MLE of α in terms of λ.

$$\hat{\alpha} = \frac{-n}{\sum_{i=1}^{n} \ln\left(\frac{\hat{\lambda}+d_i}{\hat{\lambda}+x_i}\right)} \tag{2}$$

Therefore, we can theoretically find the MLE of α by first finding the MLE of λ. Unfortunately, this is not as straight forward as it may seem. There is no finite λ that will maximise the log likelihood, as the log likelihood is always increasing with respect to λ.

$$\frac{\partial l}{\partial \lambda} = \sum_{i=1}^{n} \left[\frac{-(\alpha + 1)}{\lambda + x_i} + \frac{\alpha}{\lambda + d_i}\right]$$

Practically, the above derivative is almost always positive since $x_i \geq d_i$.
As such, we are unable to find estimates of our parameters by maximum likelihood as the likelihood

has no maximum. In fact, a higher λ will always result in a higher likelihood.
This is not too much of an issue as both the Log Normal and Gamma distribution is shown to be a much better fit anyways. For sake of providing an estimate, the MLE given are:

$$\hat{\lambda} = 200,000$$
$$\hat{\alpha} = 51.18$$

The above parameters yield a negative log likelihood of:

$$-l = 11,014.80$$

1.4 Log-Normal distribution

The Log-Normal (LN) distribution also has two parameters: μ and σ.
The PDF of a LN distribution with these parameters is

$$f(x;\mu,\sigma) = \frac{1}{\sigma\sqrt{2\pi}}\frac{1}{x}\exp\left[-\frac{1}{2}\left(\frac{\log x-\mu}{\sigma}\right)^2\right],\ x>0$$

There is no analytical solution to find the MLE so numerical techniques are used. A variety of initial estimates were tested, and all converged to an MLE estimate:

$$\hat{\mu} = 8.432$$
$$\hat{\sigma} = 0.396$$

This yields a negative log likelihood of

$$-l = 10,571.328$$

1.5 Gamma distribution

The Gamma distribution also has two parameters: shape parameter α and rate parameter λ. With density function

$$f(x;\alpha,\lambda) = \frac{\lambda^\alpha}{\Gamma(\alpha)}x^{\alpha-1}e^{-\lambda x},\ x>0$$

Similar to the LN distribution, there is no analytical solution to finding the MLE solutions. Numerical methods are used to find the estimates:

$$\alpha = 6.555$$
$$\lambda = 0.001$$

Which yields a negative log likelihood of:

$$-l = 10,576.264$$

1.6 Calculations for visual analysis

When plotting the fitted distributions to the empirical data, it is important to consider effects of the deductible. As the raw data is a combination of different claims from different deductible levels, the overlayed theoretical curves must reflect this heterogeneity. By assuming that loss severity is independent of a policy's deductible (as long as it is greater than the deductible), the appropriate curves can be calculated using a simple combination of weights and indicator functions.

$$f_D(x) = \sum_{i=1}^{|d|} w_i \times f_{d_i}(x) \times I_{\{x>d_i\}}$$
$$F_D(x) = \sum_{i=1}^{|d|} w_i \times F_{d_i}(x) \times I_{\{x>d_i\}}$$

Where:

$f_D(x) =$ Theoretical density for the raw data

$F_D(x) =$ The probability that a randomly picked loss severity from the dataset is less than or equal to x

$|d| =$ The number of deductible levels, 6 in this case: 700 (default), 950, 1200, 1400, 1700, 1900

$d_i =$ i-th deductible level

$w_i = \frac{n_{d_i}}{n} =$ The proportion of policies in the dataset with deductible level d_i

$f_{d_i}(x) = \frac{f(x)}{1 - F(d_i)} =$ The conditional density of a loss with deductible level d_i

$F_{d_i}(x) = \frac{F(x)}{1 - F(d_i)} =$ The conditional CDF of a loss with deductible d_i

$I_{\{x>d_i\}} = \begin{cases} 1 & \text{if } x > d_i \\ 0 & \text{otherwise} \end{cases}$ an indicator function

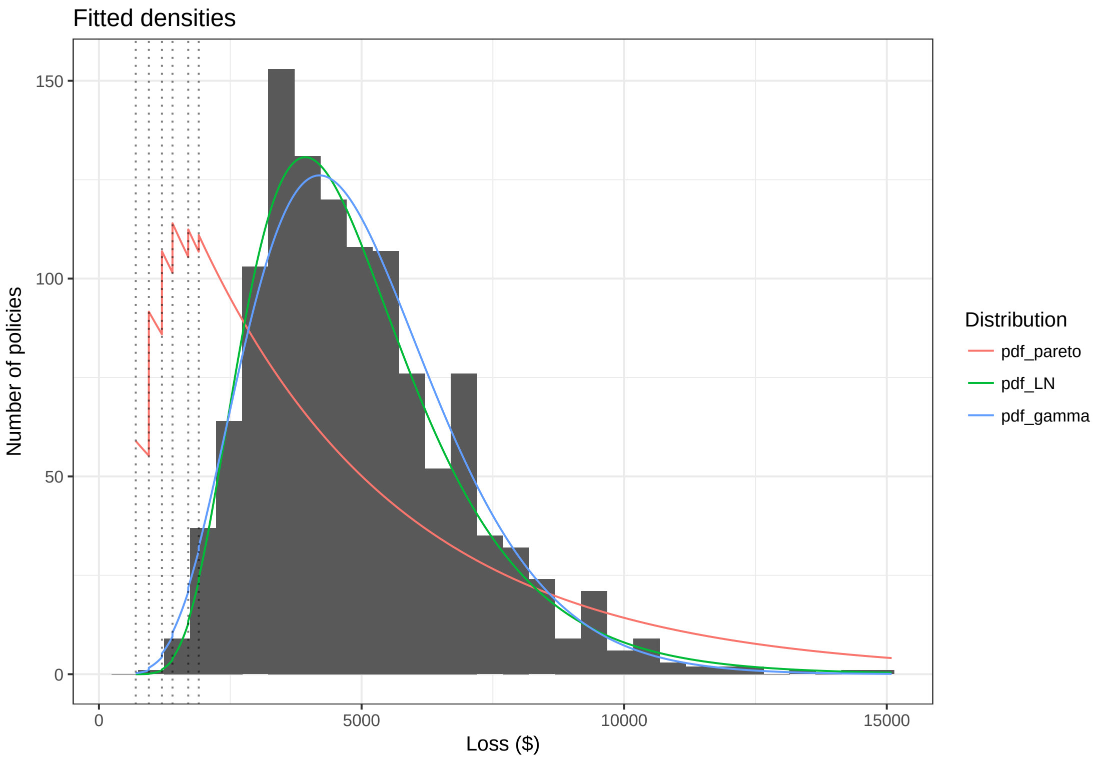


Figure 1: Empirical histogram with fitted densities overlay

In figure 1, we can clearly see that the Pareto distribution's shape is simply not a good fit at all to the observed data. Empirically the data is unimodal with tails on both sides of the peak. The data has a long right tail, also shared by the Pareto distribution. However, the Pareto distribution does not have any tail to the left of its peak, the density simply decreases monotonically. The jumps in theoretical densities is due to the different deductible levels, represented by the dotted vertical lines. Both the Log Normal and Gamma distribution fit the data much better. To differentiate between the two, it is clearer to look at the CDF.

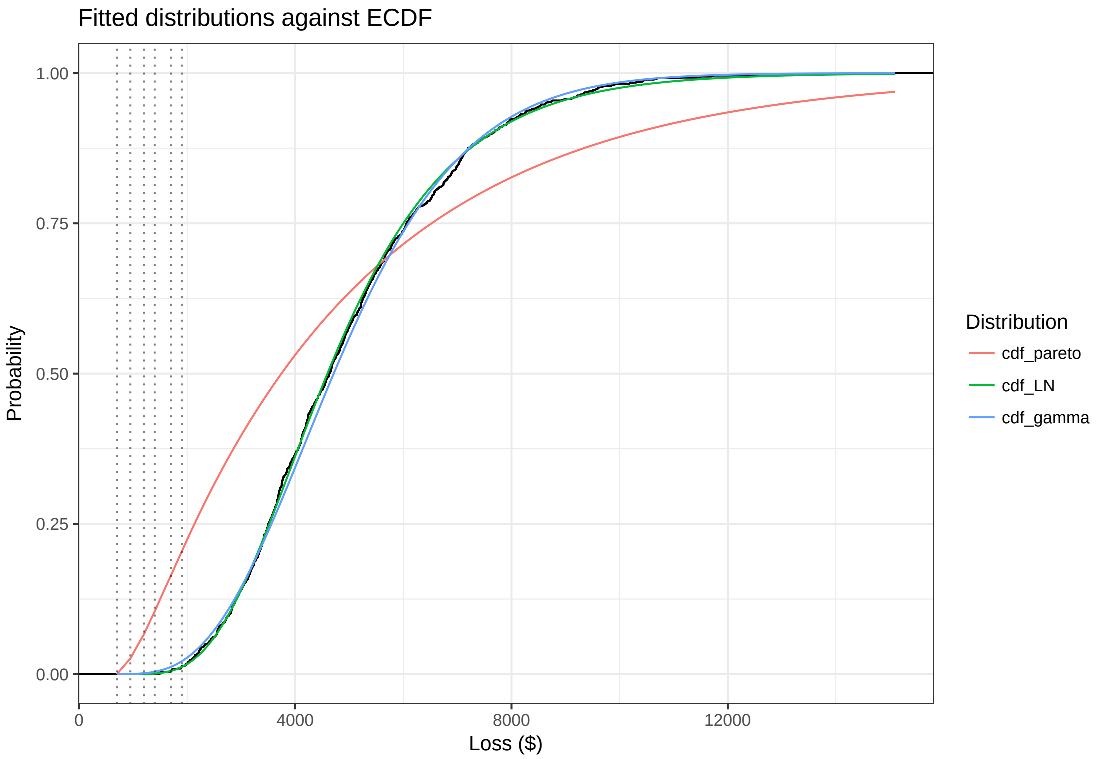
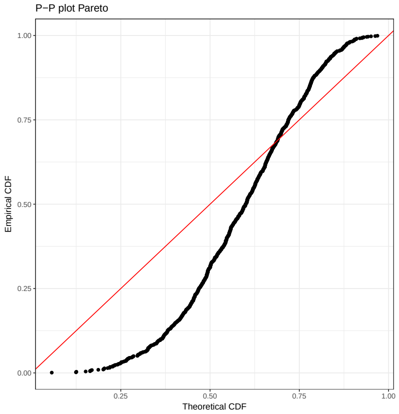


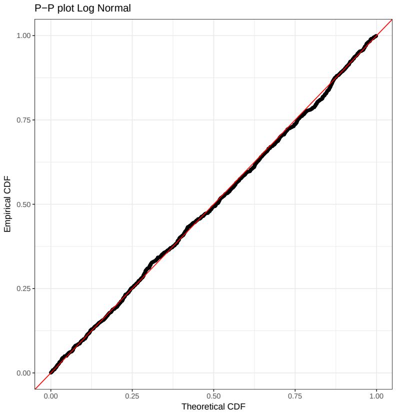
Figure 2: Empirical CDF with fitted distribution overlay

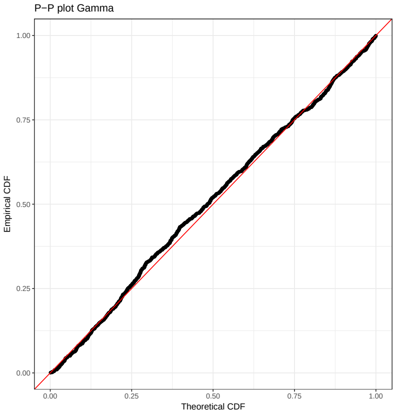
In figure 2, it is again clear to see that the Pareto distribution does not fit the data as well as the LN or Gamma. Furthermore, the green line of the LN model follows the black line of the empirical CDF slightly better. This is especially noticeable in the loss range of 4,000 - 7,000, where the Gamma distribution is slightly under the empirical CDF.

1.6.1 P-P plots

The following are P-P plots for the 3 distributions





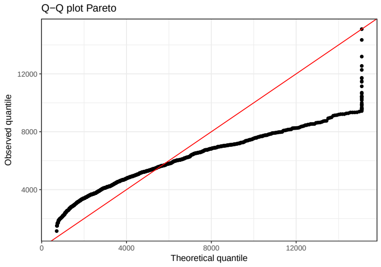


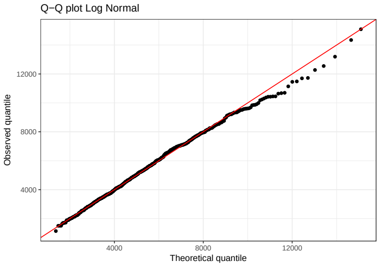
It is again clear that the Pareto distribution is not a good fit, with the points not anywhere near close to the $y = x$ line. While both the Log Normal and Gamma distribution seem to be good fits, the

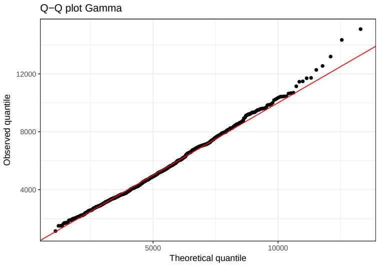
Gamma distribution seems to deviate around the middle, again suggesting that the Log Normal model is a better fit.

1.6.2 Q-Q plots

Q-Q plots are better at revealing deviations around the tails. The following are the Q-Q plots for the three distributions.







As expected, the Q-Q plot for the Pareto distribution indicates that it is not a good fit anywhere. It is much more interesting to compare the Q-Q plots between the Log Normal and Gamma distribution. Both are very good fits until the end of the right tail. The Log Normal distribution has a tail that seems too fat for the data given whereas the the Gamma distribution's tail isn't fat enough. That being said, it still appears that the observations appear to follow a Log Normal distribution better as the points move back towards the $y = x$ line for the Log Normal Q-Q plot but less so for the Gamma Q-Q plot.

Furthermore, it can be argued that a model which overestimates loss severity is better than one which underestimates. This is because an overestimation of claims cost lead to higher reserves and decreased probability of ruin. Thus the Log Normal distribution is the most appropriate distribution to model claim size.

1.7 Hypothesis tests

The Pearson's chi-squared and Kolmogorov-Smirnoff test were conducted for all three distributions. Both these tests reject the null that the data is Pareto distributed and fail to reject the null for the Log Normal and Gamma distribution. This aligns with previous drawn conclusions that the Log Normal and Gamma distributions is a good fit and the Pareto distribution is a bad fit.

Additionally, the p-values of the hypothesis tests are always bigger for the Log Normal distribution, indicating that it has the better fit.

1.7.1 Pearson's Chi-squared test

To conduct this hypothesis test, the data points must be placed in different bins. Table 1 summarises the construction of bins and the number of losses in each one.

Bin range	Number of losses
$1000 < X \leq 2000$	21
$2000 < X \leq 2500$	53
$2500 < X \leq 3000$	93
$3000 < X \leq 2500$	127
$3500 < X \leq 4000$	138
$4000 < X \leq 2500$	129
$4500 < X \leq 5000$	124
$5000 < X \leq 2500$	110
$5500 < X \leq 6000$	79
$6000 < X \leq 2500$	62
$6500 < X \leq 7000$	66
$7000 < X \leq 2500$	56
$7500 < X \leq 8000$	37
$8000 < X \leq 2500$	24
$8500 < X \leq 9000$	15
$9000 < X \leq 2500$	16
$9500 < X \leq 10000$	14
$10000 < X \leq 20000$	21

Table 1: Table of binned losses

The test statistic is calculated by

$$\chi^2 = \sum_i \frac{(O_i - E_i)^2}{E_i}$$

Where:

$$O_i = \text{Number of losses in bin } i$$
$$E_i = \text{Expected number of losses in bin } i \text{ given fitted distribution}$$

Under the null hypothesis (that the data does come from the fitted distribution), the test statistic follows a Chi-squared distribution with degrees of freedom equal to the number of bins minus the number of parameters in the fitted model minus 1:

$$\sum_i \frac{(O_i - E_i)^2}{E_i} \sim \chi^2(n - k - 1)$$

Where:

$$n = \text{Number of bins} = 18$$
$$k = \text{Number of parameters in the model} = 2$$

Distribution	Test Statistic	p value
Pareto	668.54	$< 1 \times 10^{-7}$
Log Normal	14.651	0.477
Gamma	21.233	0.130

Table 2: Results of Chi-squared test

Table 2 show the results of the hypothesis tests. There is significant evidence to show that the data is not Pareto distributed. There is not enough evidence to say that it's not log-normally distributed or Gamma distributed.

1.7.2 Kolmogorov-Smirnoff test

The Kolmogorov-Smirnoff (K-S) test looks at the biggest absolute difference between the empirical CDF and fitted CDF.

$$KS = \sup_x |F(x) - \hat{F}(x)|$$

Where:

$$F(x) = \text{The fitted CDF}$$
$$\hat{F}(x) = \text{The empirical CDF}$$

The critical values are given by table 3.

α	20%	15%	10%	5%	1%	0.1%
Formula	$\frac{1.073}{\sqrt{n}}$	$\frac{1.138}{\sqrt{n}}$	$\frac{1.224}{\sqrt{n}}$	$\frac{1.358}{\sqrt{n}}$	$\frac{1.628}{\sqrt{n}}$	$\frac{1.94947}{\sqrt{n}}$
Value	0.0312	0.0331	0.0356	0.0394	0.0473	0.0566

Table 3: Critical values of the KS test statistic, http://www.real-statistics.com/statistics-tables/kolmogorov-smirnov-table/

Distribution	Test Statistic	p value
Pareto	0.2647	< 0.001
Log Normal	0.0205	> 0.2
Gamma	0.0348	> 0.1

Table 4: Results of KS test

Table 4 shows that, under a significance level of 5%, the null hypothesis can't be rejected when fitted to the Log Normal or Gamma distribution. Again there is significant evidence suggesting that the data is not Pareto distributed.

2 Task 2

2.1 a)

Since $S_i \sim \text{Gamma}(\alpha,\, \beta)$, we have $\mathbb{E}[S_i] = \frac{\alpha}{\beta}$

$$\begin{aligned}
\psi_1(c_0) &= \text{Probability of ruin at time 1} \\
&= \Pr(C_1 < 0) \\
&= \Pr(c_0 + (\pi_1 - S_1) < 0) \\
&= \Pr\left(c_0 + 1.2\frac{\alpha}{\beta} < S_1\right) \qquad \text{since } \pi_i = \pi = 1.2\,\mathbb{E}[S] = 1.2\frac{\alpha}{\beta} \\
&= 1 - \Pr\left(S_1 \leq c_0 + 1.2\,\frac{\alpha}{\beta}\right) \\
&= 1 - G\left(c_0 + 1.2\,\frac{\alpha}{\beta};\, \alpha,\, \beta\right)
\end{aligned}$$

2.2 b)

This is an expression for the probability that ruin occurs by time 2.

$$\psi_2(c_0) = \int_0^{c_0+\pi} \psi_1(c_0 + \pi - y)\, g(y;\, \alpha,\, \beta)\, dy + 1 - G(c_0 + \pi;\, \alpha,\, \beta)$$

For the company to be ruined by time 2, one of two disjoint events must occur. Either the company is ruined by time 1, or it survives to time 1 but is ruined by time 2. The above equation is simply the sum of the probabilities of those two events.
The last two terms of the equation represent the probability that ruin will occur by the first time step. Noting that $\pi = 1.2\frac{\alpha}{\beta}$ we see that $1 - G(c_0 + \pi;\, \alpha,\, \beta) = \psi_1(c_0)$ as shown in part a). Thus this accounts for the possibility that the insurer is ruined by time step 1.

The integral represents the probability that the insurer survives to time step 1, but is ruined by time step 2. Suppose the insurer suffers of a loss of y in time step 1, this leaves them with capital $c_0 + \pi - y$ as the premium, π, stays constant. For the insurer to have survived, the value of y must be such that $c_0 + \pi - y \geq 0 \Leftrightarrow y \leq c_0 + \pi$.
Assuming that the insurer does survive to time 1, the probability that they are ruined by time 2 is then $\psi_1(c_0 + \pi - y)$. The density of this event occurring, where the insurer suffers a loss of y, survives to time 1 but is ruined by time 2, can be expressed as $\psi_1(c_0 + \pi - y)\, g(y;\, \alpha,\, \beta)$ where g is the density of the loss.
The probability can then be found by integrating over all possible values of y. Remembering that $y > 0$ and $y \leq c_0 + \pi$, the appropriate integral is then:

$$\underbrace{\int_0^{c_0+\pi}}_{\text{all possible values of } y} \qquad \overbrace{\psi_1(c_0 + \pi - y)}^{\text{ruin at time 2 given loss of } y} \qquad \underbrace{g(y;\, \alpha,\, \beta)}_{\text{density of loss equaling } y} \qquad dy$$

Summing these two probabilities together yields

$$\psi_2(c_0) = \underbrace{\int_0^{c_0+\pi} \psi_1(c_0 + \pi - y)\, g(y;\, \alpha,\, \beta)\, dy}_{\text{Probability of surviving to time 1 and ruin at 2}} + \underbrace{1 - G(c_0 + \pi;\, \alpha,\, \beta)}_{\text{Probability of ruin at time 1}}$$

Assuming $\alpha = 20$ and $\beta = 0.2$, the following was calculated:

c_0	$\psi_2(c_0)$
0	0.2177
10	0.1259
20	0.0680
50	0.0078

2.3 c)

The difficulty in calculating $\psi_t(c_0)$ at higher values of t is the number of calculations involved. These integrals do not have closed form solutions. As a result, numerical methods must be used. For higher values of t, it is necessary to take many integrals of integrals, greatly increasing the number of calculations required. Additionally, $g(y;\,\alpha,\,\beta)$ contains $\Gamma(\alpha)$ which must also be calculated using numerical methods.
All this contribute to the rapidly increasing computation complexity of such a calculation and may require many resources.

2.4 d)

- **Give an expression for $\psi_1^*(c_0)$.**

$$\begin{aligned}\psi_1^*(c_0) &= \Pr(C_1^* < 0)\\ &= \Pr(c_0 + \pi_1 + S_1^* < 0)\\ &= \Pr(S_1^* > c_0 + \pi)\\ &= 1 - \Pr(S_1^* \leq c_0 + \pi)\\ &= 1 - G^*(c_0 + \pi)\end{aligned}$$

- **Provide a recursive expression for $\psi_t^*(c_0)$**
  Using a similar logic to part b),

$$\psi_t^*(c_0) = \psi_1^*(c_0) + \sum_{y=0}^{\lfloor c_0+\pi \rfloor} g^*(y)\,\psi_{t-1}^*(c_0 + \pi - y)$$

  Note that the first term represents the probability of immediate ruin. The summation is the probability of ruin by time t given that the company survives time 1. The summation sums over all possible loss values that won't initially ruin the company, from $y = 0$ to $c_0 - \pi$, the largest integer such that the capital $c_0 + \pi - y$ will still be non-negative

- **Comment on the implementation of this expression in comparison to the original case when S_i was continuous.**
  The implementation of this algorithm require a much lower number of calculations. This is because of a few reasons.
  Firstly, $g^*(y)$ is calculated beforehand and stored. The comparable term in the continuous version must be numerically calculated.
  Secondly, each value of $\psi_{t-1}^*(c_0 + \pi - y)$ can be stored and re-used in later iterations of the algorithm. Conversely, the continuous ψ in the exact algorithm may need to be evaluated from scratch at each iteration.
  Lastly, the summation contains $\lfloor c_0 + \pi \rfloor + 1$ terms. Contrastingly, the integral in the continuous recursion must be solved numerically, and so will involve a much larger number of calculations.
  The implementation by discretisation of S will drastically reduce the computational effort involved. This comes at a cost of accuracy, but for appropriate discretisation, the discrepancies can be minimal and insignificant.

2.5 e)

2.5.1 (A)

Method of rounding

1. $h = 1$, $m = 150$

c_0	$\psi_2^*(c_0)$
0	0.2121
10	0.1222
20	0.0659
50	0.0520

2. $h = 1$, $m = 300$

c_0	$\psi_2^*(c_0)$
0	0.2121
10	0.1222
20	0.0659
50	0.0075

3. $h = 5$, $m = 30$

c_0	$\psi_2^*(c_0)$
0	0.1911
10	0.1086
20	0.0578
50	0.0444

4. $h = 5$, $m = 60$

c_0	$\psi_2^*(c_0)$
0	0.1911
10	0.1086
20	0.0578
50	0.0065

2.5.2 (B)

Method of upper and lower bounds

1. d = 1

c_0	$\psi_2^*(c_0)$ Lower	$\psi_2^*(c_0)$ Upper
0	0.2190	0.2055
10	0.1269	0.1177
20	0.0688	0.0631
50	0.0080	0.0071

2. d = 5

c_0	$\psi_2^*(c_0)$ Lower	$\psi_2^*(c_0)$ Upper
0	0.2245	0.1616
10	0.1314	0.0892
20	0.0721	0.0461
50	0.0087	0.0048

2.5.3 Compare and comment on differences in your results relative to part b).

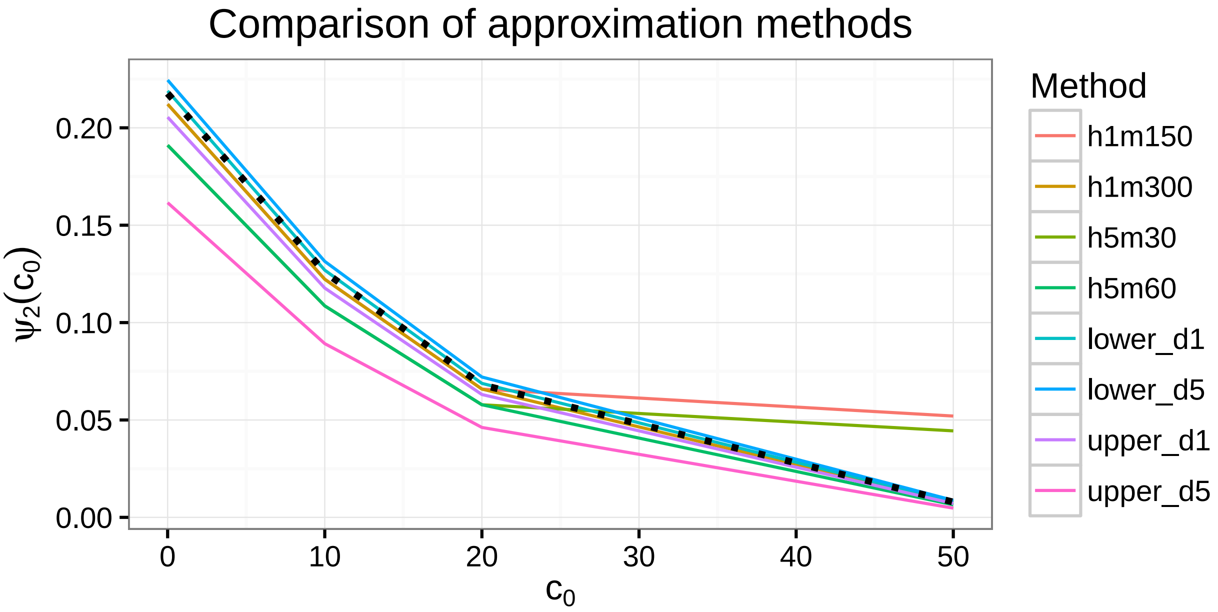


Figure 3: Comparison of how approximation methods effect the probability of ruin

The probability of ruin calculated under each approximation method is plotted above along with the true value (dotted black line). There are several key features to note about the different approximation methods.

- Method of rounding
  - Lower values of h yielded a better approximation
    This is because the level of granularity is higher with smaller values of h. A higher number of discrete mass points are being used to approximate a continuous distribution.
  - Approximations with the same h but different m yield the same results up to a certain point. The point where they differ comes when $h \times m$, the end value of the approximation, doesn't hold enough information to give a good approximation.
    This is clearly shown by the orange and green line deviating from the pack from $c_0 = 20$ to $c_0 = 50$.
    When the initial capital is 50, the loss severity needed to ruin the company increases, but when $h \times m = 150$, the discrete approximation doesn't hold enough information about the tail end of the distribution to approximate the true probability well.
- Method of upper and lower bounds
  - $\psi_2^*(c_0)$ is always smaller than the true probability when using the upper bound approximation and vice versa
  - The approximation is better with smaller values of d. This is because it provides a finer level of detail resulting in more accurate probabilities

3 Task 3

3.1 a)

3.1.1 Distribution of Y_5

Use the method of moment generating functions (mgf) to find the distribution of Y_5. Each distribution's mgf is unique to itself, and it can be shown that

$$Y_5 \sim \text{Gamma}(25,\ 0.8)$$

Proof:

$$\begin{aligned}
M_{Y_5}(t) &= \mathbb{E}\left[e^{Y_5\, t}\right] \\
&= \mathbb{E}\left[e^{t\, \sum_{i=1}^{5} S_i}\right] \\
&= \mathbb{E}\left[\prod_{i=1}^{5} e^{t\, S_i}\right] \\
&= \prod_{i=1}^{5} \mathbb{E}\left[e^{t\, S_i}\right] \quad \text{Assuming } S_i\text{'s are independent of each other} \\
&= \prod_{i=1}^{5} M_S(t) \\
&= \prod_{i=1}^{5} \left(1 - \frac{t}{\beta}\right)^{\alpha} \\
&= \left(1 - \frac{t}{\beta}\right)^{5\alpha}
\end{aligned}$$

Which is the mgf of a Gamma distribution with parameters $5\alpha = 25$, $\beta = 0.8$. Hence

$$Y_5 \sim Gamma(25, 0.8)$$

3.1.2 Distribution function at $x = 10, 20, \ldots, 50$

This is calculated numerically through R.

x	$F_{Y_5}(x)$
10	$1.172{\times}10^{-6}$
20	0.0223
30	0.4460
40	0.9119
50	0.9955

3.2 b)

3.2.1 Discretise S_i and use Panjer's recursion to approximate Y_5

Let $F^*_{Y_5}(x)$ denote the CDF calculated using Panjer's recursion and approximated by a discretised S_i.

x	$F^*_{Y_5}(x)$
10	$1.419{\times}10^{-5}$
20	0.0324
30	0.4837
40	0.9226
50	0.9961

3.2.2 Methodology and assumptions

The discretisation of S_i to S_i^* is done by the method of rounding using parameters $h = 1$ and $m = 70$. Let g_s represent the discretised mass function. Then g_s is calculated as

$$g_s = \begin{cases} 0 & \text{if } s \text{ is not a multiple of } h \\ 0 & \text{if } s < 0 \\ 1 - G(s - \frac{h}{2}) & \text{if } s = hm \\ G(s + \frac{h}{2}) - G(s - \frac{h}{2}) & \text{otherwise} \end{cases}$$

Where G is the CDF of S_i
m was chosen to be 70 as it was big enough to capture almost all the information of the PDF of S_i. This is because $G(70) = \Pr(S_i \leq 70) \approxeq 1$.

To sum five S_i variables, it is assumed that the number of accidents, N, follows a binomial distribution with a very high probability parameter.

$$N \sim Bin(5, 0.997)$$

A binomial distribution is a member of the (a,b) class of distributions, meaning if $N \sim Bin(n,\, p)$ and $p_r = \Pr(N = r)$, then:

$$p_r = \left(a + \frac{b}{r}\right) p_{r-1}$$

Where:

$$a = \frac{-p}{1-p}$$
$$b = \frac{(n+1)p}{1-p}$$

If $p = 1.$ then N would be a constant. However Panjer recursion could not be used as the variables a, b would be undefined. $p = 0.997$ is a close approximation to a constant, however, with only a 0.015 probability of not evaluating to equal 5. With higher values of p, Panjer's recursion is not stable.

Let $f(s) = \Pr(Y_5^* = 5)$ where $Y_5^* = \sum_{i=1}^{N} S^*$ the sum of the discretised losses with frequency $N \sim Bin(n,\, p)$.
Starting with

$$\begin{aligned} f(0) &= M_N(\ln(g_0)) \\ &= (q + p \times g_0)^n \\ &= 2.688 \times 10^{-13} \\ &\approx 0 \end{aligned}$$

The values of $f(s)$, $s = 2,\, 3, \ldots$ are recursively calculated using the formula

$$f(s) = \frac{1}{1 - ag_0} \sum_{j=1}^{s} \left(a + b\frac{j}{s}\right) g_j f(s-j)$$

The distribution function is then simply

$$F_{Y_5}(s) = \sum_{i=0}^{s} f(i)$$

Assumptions:

- Each loss is iid
- N is a good enough approximation to 5
- The discretisation is a good approximation to the continuous variable
- The loss can take the value of 0.
  Although this does not reflect the original distribution, the discretised mass at 0 is insignificantly small with $g_0 < 0.0001$.
  Despite a simple correction that could have been made, it would not have had a material difference. Additionally, $f(0)$ would had to have been non-zero anyways as $\Pr(N = 0) > 0$.

3.3 Comparison

Table 5 compares the approximation to the true distribution side by side. The approximation always overestimates the true distribution value. However it is a small overestimate, with the largest only being a couple percent higher than true value.

x	$F^*_{Y_5}(x)$	$F_{Y_5}(x)$	Difference
10	1.419×10^{-5}	1.172×10^{-6}	1.3×10^{-5}
20	0.0324	0.0223	0.0101
30	0.4837	0.4460	0.0377
40	0.9226	0.9119	0.0107
50	0.9961	0.9955	0.0006

Table 5: Comparison of approximation and true value

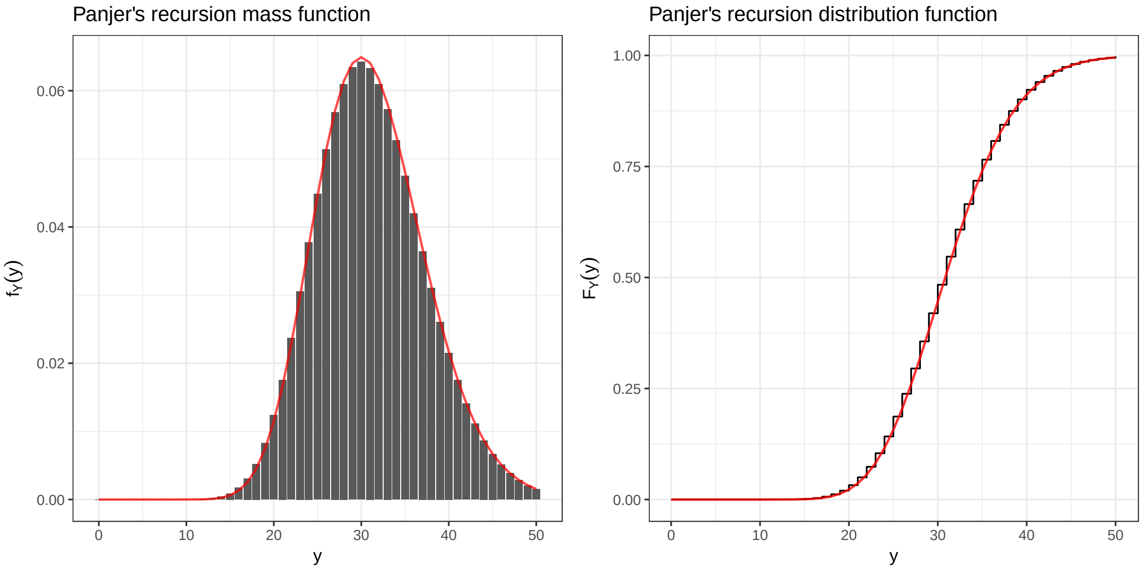


Figure 4: Comparison of approximation using Panjer recursion and discretisation to true probabilities

Figure 4 shows the approximated mass and distribution functions in grey and the true density and distribution in red. Visually, the approximation seems to follows the true values very closely.

A Task 1

A.1 MLE for Pareto

The Pareto distribution with shape parameter α, scale parameter λ have the following distributions:

$$\begin{aligned} f(x) &= \frac{\alpha\lambda^\alpha}{(\lambda+x)^{\alpha+1}}, \quad x>0 \\ F(x) &= 1-\left(\frac{\lambda}{\lambda+x}\right)^\alpha, \quad x>0 \end{aligned}$$

From equation (1) we know that

$$\begin{aligned} l &= \sum_{i=1}^{n} [\ln(f(x_i)) - \ln(1-F(d_i))] \\ &= \sum_{i=1}^{n} \left[\ln\left(\frac{\alpha\lambda^\alpha}{(\lambda+x_i)^{\alpha+1}}\right) - \ln\left(\left(\frac{\lambda}{\lambda+d_i}\right)^\alpha\right)\right] \\ &= \sum_{i=1}^{n} [\ln(\alpha) - (\alpha+1)\ln(\lambda+x_i) + \alpha\ln(\lambda+d_i)] \end{aligned}$$

Differentiating with respect to α and λ yields:

$$\frac{\partial l}{\partial \alpha} = \frac{n}{\alpha} + \sum_{i=1}^{n} \ln\left(\frac{\lambda+d_i}{\lambda+x_i}\right) \tag{3}$$

$$\frac{\partial l}{\partial \lambda} = \sum_{i=1}^{n} \left[\frac{-(\alpha+1)}{\lambda+x_i} + \frac{\alpha}{\lambda+d_i}\right] \tag{4}$$

We set the first derivatives to 0 and solve to to find the MLE:

$$\begin{aligned} \frac{\partial l}{\partial \alpha} &= 0 \\ \Leftrightarrow 0 &= \frac{n}{\hat{\alpha}} + \sum_{i=1}^{n} \ln\left(\frac{\hat{\lambda}+d_i}{\hat{\lambda}+x_i}\right) \\ \Leftrightarrow \hat{\alpha} &= \frac{-n}{\sum_{i=1}^{n} \ln\left(\frac{\hat{\lambda}+d_i}{\hat{\lambda}+x_i}\right)} \qquad (5) \\ & \qquad (6) \end{aligned}$$

B R code

B.1 Data pre processing

```
# Imports data and saves as an R image
setwd("C:/Users/johnn/Dropbox/Uni/2017␣S2/ACTL3162/Assignment/Codes")

claims <- read.csv(file = "data.csv.csv")

# General data manipulation
```

```
default_excess <- 700
x <- claims$paid
claims$excess <- default_excess + claims$elect_XS
claims$loss <- claims$paid + claims$excess
n <- nrow(claims)

save.image(file = "claims.RData")
```

B.2 Task 1

```
library(stats4)
library(ggplot2)
library(dplyr)
library(actuar)
library(reshape2)

# Clear everything to prevent unintentional reuse of objects
rm(list=ls())

# Set work directory, import data, and include relevant libraries
setwd("C:/Users/johnn/Dropbox/Uni/2017␣S2/ACTL3162/Assignment/Codes")
report_path <-
  "C:/Users/johnn/Dropbox/Uni/2017␣S2/ACTL3162/Assignment/Report/images/"
load("claims.RData")
output <- FALSE


####### Summarise by excess level #######
claims_XS_mean <- claims %>%
  group_by(excess) %>%
  summarise(mean = mean(loss),
            policies = n()) %>%
  mutate(proportion = policies/sum(policies)) %>%
  arrange(desc(policies))

###### PARETO #############

# Function to calculate MLE alpha given a lambda
f_alpha <- function(lambda){
  return(-n/sum(log((lambda + claims$excess)/(lambda + claims$loss))))
}

## Create (negative) log likelihood function
f.nLL_pareto <- function(alpha, lambda){
  l <- sum(log(dpareto(claims$loss, alpha, lambda)) -
             log(1-ppareto(claims$excess, alpha, lambda)))
  return(-l)
}

init_lambda_pareto <- 200000

mle_pareto <- mle(f.nLL_pareto,
                  start = list(alpha = f_alpha(init_lambda_pareto),
                               lambda = init_lambda_pareto))

####### LOG NORMAL ########
# Function to calculuate the negative log likelihood
f.nLL_LN <- function(mu, sigma){
  return(-sum(log(dlnorm(claims$loss, mu, sigma)) -
                log(1-plnorm(claims$excess, mu, sigma))))
}

mle_LN <- mle(f.nLL_LN, start = list(mu = 9, sigma =0.2))
mle_LN

###### GAMMA #############
# function to calculate the negative log likelihood
f.nLL_gamma <- function(alpha, lambda){
  return(-sum(log(dgamma(claims$loss, shape = alpha, rate = lambda)) -
         log(1- pgamma(claims$excess, shape = alpha, rate = lambda))))
}

mean_loss <- mean(claims$loss)
alpha_MME <- n*mean_loss^2/sum((claims$loss - mean_loss)^2)
lambda_MME <- alpha_MME/mean_loss

mle_gamma <- mle(f.nLL_gamma, start = list(alpha = alpha_MME,
                                           lambda = lambda_MME),
      control=list(ndeps=c(1e-6, 1e-6)))


######################### Plotting ########################
##########################################################

## Plotting PDF
f_pareto <- function(x, alpha, lambda){
  i <- 1:nrow(claims_XS_mean)
  prop <- claims_XS_mean$proportion
  deductible <- claims_XS_mean$excess
  result <- sum(prop[i]*dpareto(x, shape = alpha, scale = lambda)*
                  ifelse(x>=deductible, 1, 0)/
                  (1-ppareto(deductible[i], shape = alpha, scale = lambda)))
  return(result)
}

f_LN <- function(x, mu, sigma){
  i <- 1:nrow(claims_XS_mean)
  prop <- claims_XS_mean$proportion
  deductible <- claims_XS_mean$excess
  result <- sum(prop[i]*dlnorm(x, mu, sigma)*
                  ifelse(x>=deductible, 1, 0)/
                  (1-plnorm(deductible[i], mu, sigma)))
  return(result)
```

```
}

f_gamma <- function(x, alpha, lambda){
  i <- 1:nrow(claims_XS_mean)
  prop <- claims_XS_mean$proportion
  deductible <- claims_XS_mean$excess
  result <- sum(prop[i]*dgamma(x, shape = alpha, rate = lambda)*
        ifelse(x>=deductible, 1, 0)/
        (1-pgamma(deductible[i], shape = alpha, rate = lambda)))
  return(result)
}

x <- claims_XS_mean$excess[1]:ceiling(max(claims$loss))
y_pareto <- x
y_gamma <- x
y_LN <- x
for (ii in 1:length(x)){
  y_pareto[ii] <- f_pareto(x[ii],
                           coef(mle_pareto)[1],
                           coef(mle_pareto)[2])
  y_gamma[ii] <- f_gamma(x[ii],
                         coef(mle_gamma)[1],
                         coef(mle_gamma)[2])
  y_LN[ii] <- f_LN(x[ii],
                   coef(mle_LN)[1],
                   coef(mle_LN)[2])
}


fitted_pdf <- data.frame(loss = x,
                         pdf_pareto = 550000*y_pareto,
                         pdf_LN = 550000*y_LN,
                         pdf_gamma = 570000*y_gamma)
fitted_pdf_melted <- fitted_pdf %>%
  melt(id = c("loss"), variable.name = "Distribution")


plot_PDF_comp <- ggplot() + geom_histogram(data = claims, aes(loss)) +
  geom_line(data = fitted_pdf_melted, aes(x = loss, y = value,
                                          color=Distribution)) +
  labs(x = "Loss␣($)", y = "Number␣of␣policies",
       title = "Fitted␣densities")+theme_bw() +
  geom_vline(xintercept = claims_XS_mean$excess, alpha = 0.5,
             linetype = "dotted") + theme_bw()


# Plotting CDF
F_pareto <- function(x, alpha, lambda){
  i <- 1:nrow(claims_XS_mean)
  prop <- claims_XS_mean$proportion
  deductible <- claims_XS_mean$excess
  result <- sum(prop * (ppareto(x, shape = alpha, scale = lambda) -
                  ppareto(deductible, shape = alpha, scale = lambda))*
                  ifelse(x>=deductible, 1, 0) /
                  (1-ppareto(deductible, shape = alpha, scale = lambda)))
  return(result)
}

F_LN <- function(x, mu, sigma){
  i <- 1:nrow(claims_XS_mean)
  prop <- claims_XS_mean$proportion
  deductible <- claims_XS_mean$excess
  result <- sum(prop*(plnorm(x, mu, sigma)-plnorm(deductible, mu, sigma))*
                  ifelse(x>=deductible, 1, 0)/
                  (1-plnorm(deductible, mu, sigma)))
  return(result)
}

F_gamma <- function(x, alpha, lambda){
  i <- 1:nrow(claims_XS_mean)
  prop <- claims_XS_mean$proportion
  deductible <- claims_XS_mean$excess
  result <- sum(prop*(pgamma(x, shape = alpha, rate = lambda) -
                        pgamma(deductible, shape = alpha, rate = lambda)) *
                  ifelse(x>=deductible, 1, 0)/
                  (1-pgamma(deductible, shape = alpha, rate = lambda)))
  return(result)
}

Y_pareto <- x
Y_gamma <- x
Y_LN <- x
for (ii in 1:length(x)){
  Y_pareto[ii] <- F_pareto(x[ii],
                           coef(mle_pareto)[1],
                           coef(mle_pareto)[2])
  Y_gamma[ii] <- F_gamma(x[ii],
                         coef(mle_gamma)[1],
                         coef(mle_gamma)[2])
  Y_LN[ii] <- F_LN(x[ii],
                   coef(mle_LN)[1],
                   coef(mle_LN)[2])

}


fitted_cdf <- data.frame(loss = x,
                         cdf_pareto = Y_pareto,
                         cdf_LN = Y_LN,
                         cdf_gamma = Y_gamma)
fitted_cdf_melted <- fitted_cdf %>%
  melt(id = c("loss"), variable.name = "Distribution")
```

```
names(fitted_cdf)[names(fitted_cdf)=="cdf_pareto"] <- "Pareto"
names(fitted_cdf)[names(fitted_cdf)=="cdf_LN"] <- "Log␣Normal"
names(fitted_cdf)[names(fitted_cdf)=="cdf_gamma"] <- "Gamma"

plot_CDF_comp <- ggplot() + stat_ecdf(data = claims, aes(loss)) +
  geom_line(data = fitted_cdf_melted, aes(x = loss, y = value,
                                          color=Distribution)) +
  labs(x = "Loss␣($)", y = "Probability",
       title = "Fitted␣distributions␣against␣ECDF") +
  geom_vline(xintercept = claims_XS_mean$excess, alpha = 0.5,
             linetype = "dotted") + theme_bw()

if (output){
  ggsave(paste(report_path, "T1_PDF_comparison.pdf", sep = ""), plot_PDF_comp)
  ggsave(paste(report_path, "T1_CDF_comparison.pdf", sep = ""), plot_CDF_comp)
}

############# PP plot #################################
ordered_losses <- claims %>%
  select(loss) %>%
  arrange(loss) %>%
  mutate(theo_pareto = 0,
         theo_LN = 0,
         theo_gamma = 0)

ordered_losses$ecdf <- (1:n)/(n+1)
for (i in 1:n){
  ordered_losses$theo_pareto[i] <- F_pareto(ordered_losses$loss[i],
                                          coef(mle_pareto)[1],
                                          coef(mle_pareto)[2])
  ordered_losses$theo_LN[i] <- F_LN(ordered_losses$loss[i],
                                            coef(mle_LN)[1],
                                            coef(mle_LN)[2])
  ordered_losses$theo_gamma[i] <- F_gamma(ordered_losses$loss[i],
                                            coef(mle_gamma)[1],
                                            coef(mle_gamma)[2])
}

pp_pareto <- ggplot(data = ordered_losses, aes(x = theo_pareto, y = ecdf)) +
  geom_point() +
  geom_abline(slope = 1, intercept = 0, col = "red") +
  labs(x = "Theoretical␣CDF", y = "Empirical␣CDF", title = "P-P␣plot␣Pareto") +
  theme_bw()
pp_LN <- ggplot(data = ordered_losses, aes(x = theo_LN, y = ecdf)) +
  geom_point() +
  geom_abline(slope = 1, intercept = 0, col = "red") +
  labs(x = "Theoretical␣CDF", y = "Empirical␣CDF", title = "P-P␣plot␣Log␣Normal") +
  theme_bw()
pp_gamma <- ggplot(data = ordered_losses, aes(x = theo_gamma, y = ecdf)) +
  geom_point() +
  geom_abline(slope = 1, intercept = 0, col = "red") +
  labs(x = "Theoretical␣CDF", y = "Empirical␣CDF", title = "P-P␣plot␣Gamma") +
  theme_bw()
if (output){
  ggsave(paste(report_path, "PP_pareto.pdf", sep = ""), pp_pareto)
  ggsave(paste(report_path, "PP_LN.pdf", sep = ""), pp_LN)
  ggsave(paste(report_path, "PP_gamma.pdf", sep = ""), pp_gamma)
}

############# Q-Q plot ################################
merged_df <- ordered_losses %>% select(loss, ecdf) %>%
  merge(fitted_cdf %>% rename(quantile = loss), by = NULL)%>%
  mutate(diff_pareto = abs(ecdf - Pareto),
         diff_LN = abs(ecdf - `Log Normal`),
         diff_gamma = abs(ecdf - Gamma))

pareto_quantiles <- merged_df %>% select(loss, ecdf, diff_pareto) %>%
  group_by(loss, ecdf) %>%
  summarise(diff_pareto = min(diff_pareto))

pareto_quantiles_1 <- merge(pareto_quantiles,
                            merged_df %>% select(loss, quantile, diff_pareto),
                            by = c("diff_pareto", "loss"))

LN_quantiles <- merged_df %>% select(loss, ecdf, diff_LN) %>%
  group_by(loss, ecdf) %>%
  summarise(diff_LN = min(diff_LN))

LN_quantiles_1 <- merge(LN_quantiles,
                        merged_df %>% select(loss, quantile, diff_LN),
                        by = c("diff_LN", "loss"))

gamma_quantiles <- merged_df %>% select(loss, ecdf, diff_gamma) %>%
  group_by(loss, ecdf) %>%
  summarise(diff_gamma = min(diff_gamma))

gamma_quantiles_1 <- merge(gamma_quantiles,
                           merged_df %>% select(loss, quantile, diff_gamma),
                           by = c("diff_gamma", "loss"))


pareto_qq <- ggplot(data = pareto_quantiles_1, aes(x = quantile, y = loss)) +
  geom_point() + geom_abline(intercept = 0, slope = 1, col = "red") +
  labs(title="Q-Q␣plot␣Pareto", x = "Theoretical␣quantile", y = "Observed␣quantile") +
  theme_bw()

LN_qq <- ggplot(data = LN_quantiles_1, aes(x = quantile, y = loss)) +
  geom_point() + geom_abline(intercept = 0, slope = 1, col = "red") +
  labs(title="Q-Q␣plot␣Log␣Normal", x = "Theoretical␣quantile", y = "Observed␣quantile")
  theme_bw()
```

```
gamma_qq <- ggplot(data = gamma_quantiles_1, aes(x = quantile, y = loss)) +
  geom_point() + geom_abline(intercept = 0, slope = 1, col = "red") +
  labs(title="Q-Q␣plot␣Gamma", x = "Theoretical␣quantile", y = "Observed␣quantile") +
  theme_bw()

if (output){
  ggsave(paste(report_path, "QQ_pareto.pdf", sep = ""), pareto_qq)
  ggsave(paste(report_path, "QQ_LN.pdf", sep = ""), LN_qq)
  ggsave(paste(report_path, "QQ_gamma.pdf", sep = ""), gamma_qq)
}


########### Comparing statistics ######################
distribution_summary_stat <-
  data.frame(Measure = c("Parameter␣1␣estimate",
                        "Parameter␣2␣estimate",
                        "Negative␣log␣likelihood",
                        "AIC",
                        "BIC"),
            Pareto = c(coef(mle_pareto)[1],
                       coef(mle_pareto)[2],
                       f.nLL_pareto(coef(mle_pareto)[1],
                                    coef(mle_pareto)[2]),
                       AIC(mle_pareto),
                       AIC(mle_pareto, k = log(n))) %>%
              round(3),
            Log.Normal = c(coef(mle_LN)[1],
                             coef(mle_LN)[2],
                             f.nLL_LN(coef(mle_LN)[1],
                                          coef(mle_LN)[2]),
                             AIC(mle_LN),
                             AIC(mle_LN, k = log(n)))%>%
              round(3),
            Gamma = c(coef(mle_gamma)[1],
                      coef(mle_gamma)[2],
                      f.nLL_gamma(coef(mle_gamma)[1],
                                   coef(mle_gamma)[2]),
                      AIC(mle_gamma),
                      AIC(mle_gamma, k = log(n)))%>%
              round(3))

############################### Hypothesis tests ###########
##################### Chi-squared #########################
# Choose bin points
bin_endpoints <- c(1000, seq(from = 2000, to = 10000, by = 500), c(20000))

# degrees of freedom = #bins - #parameters - 1
df <- (length(bin_endpoints) - 1) - 2 - 1
# Put losses into bins
grouped_losses <- claims %>%
  select(loss)
```

```
grouped_losses$binned <- cut(grouped_losses$loss, bin_endpoints)

# Count the number in each bin
grouped_losses_count <- grouped_losses %>%
  group_by(binned) %>%
  summarise(observations = n()) %>%
  mutate(pareto_theoretical = 0,
         ln_theoretical = 0,
         gamma_theoretical = 0)

# Compare with theoretical proportion in each bin
for (row in 1:nrow(grouped_losses_count)){
  # Pareto
  grouped_losses_count$pareto_theoretical[row] <- F_pareto(bin_endpoints[row + 1],
                                                           coef(mle_pareto)[1],
                                                           coef(mle_pareto)[2]) -
    F_pareto(bin_endpoints[row],
             coef(mle_pareto)[1],
             coef(mle_pareto)[2])

  # Log-Normal
  grouped_losses_count$ln_theoretical[row] <- F_LN(bin_endpoints[row + 1],
                                                           coef(mle_LN)[1],
                                                           coef(mle_LN)[2]) -
    F_LN(bin_endpoints[row],
         coef(mle_LN)[1],
         coef(mle_LN)[2])

  # Gamma
  grouped_losses_count$gamma_theoretical[row] <- F_gamma(bin_endpoints[row + 1],
                                                   coef(mle_gamma)[1],
                                                   coef(mle_gamma)[2]) -
    F_gamma(bin_endpoints[row],
            coef(mle_gamma)[1],
            coef(mle_gamma)[2])
}

# Find chi - squared test statistic for each distribution
X2_hypothesis_test <- grouped_losses_count %>%
  mutate(X2_pareto = (observations -
                        n * pareto_theoretical)^2/(n * pareto_theoretical),
         X2_ln = (observations - n * ln_theoretical)^2/(n * ln_theoretical),
         X2_gamma = (observations - n * gamma_theoretical)^2/
           (n * gamma_theoretical)) %>%
  select(X2_pareto, X2_ln, X2_gamma) %>%
  melt(variable.name = "Distribution") %>%
  group_by(Distribution) %>%
  summarise(Test_statistic = sum(value)) %>%
  mutate(p_value = 1 - pchisq(Test_statistic, df))
```

```
################## Kolmogorov Smirnoff #################
loss_ecdf <- ecdf(claims$loss)

### Pareto
KS_hypothesis_test <- data.frame(loss = claims$loss) %>%
  mutate(Pareto = 0,
         ln = 0,
         gamma = 0)

for (row in 1:nrow(KS_hypothesis_test)){
  KS_hypothesis_test$Pareto[row] <- abs(loss_ecdf(KS_hypothesis_test$loss[row]) -
                                          F_pareto(KS_hypothesis_test$loss[row],
                                                   coef(mle_pareto)[1],
                                                   coef(mle_pareto)[2]))

  KS_hypothesis_test$ln[row] <- abs(loss_ecdf(KS_hypothesis_test$loss[row]) -
                                          F_LN(KS_hypothesis_test$loss[row],
                                                   coef(mle_LN)[1],
                                                   coef(mle_LN)[2]))

  KS_hypothesis_test$gamma[row] <- abs(loss_ecdf(KS_hypothesis_test$loss[row]) -
                                          F_gamma(KS_hypothesis_test$loss[row],
                                                   coef(mle_gamma)[1],
                                                   coef(mle_gamma)[2]))
}

KS_statistics <- KS_hypothesis_test %>%
  select(-loss) %>%
  melt(variable.name = "Distribution") %>%
  group_by(Distribution) %>%
  summarise(KS_test_statistic = max(value))

# Critical values for alpha = 20%, 15%, 10%, 5%, 1%, 0.1%
KS_critical <- c(1.073, 1.138, 1.224, 1.358, 1.628, 1.94947)/sqrt(n)
KS_critical
```

B.3 Task 2

```
library(dplyr)
library(actuar)
library(ggplot2)
library(reshape2)

# Clear everything to prevent unintentional reuse of objects
rm(list=ls())

# Export parameters
report_path <-
  "C:/Users/johnn/Dropbox/Uni/2017␣S2/ACTL3162/Assignment/Report/images/"
export <- FALSE

# Loss distribution parameters
```

```
alpha = 20
beta = 0.2
v_c0 = c(0, 10, 20, 50)
premium = 1.2 * alpha / beta

# Initialise dataframe to store approximations
all_psi2 <- data.frame(c0 = v_c0,
                       actual = rep(0, length(v_c0)),
                      h1m150 = rep(0, length(v_c0)),
                      h1m300 = rep(0, length(v_c0)),
                      h5m30 = rep(0, length(v_c0)),
                      h5m60 = rep(0, length(v_c0)))  %>%
  mutate(lower_d1 = 0,
         lower_d5 = 0,
         upper_d1 = 0,
         upper_d5 = 0)

row.names(all_psi2) <- v_c0
for (init_capital in v_c0) {
  c0 <- init_capital
  integrand <- function(y){
    (1 - pgamma(c0 + 2 * premium - y, shape = alpha, rate = beta)) *
      dgamma(y, shape = alpha, rate = beta)
  }
  integral_val <- integrate(integrand, lower = 0, upper = c0 + premium)$value
  psi_2 <- integral_val + 1 - pgamma(c0 + premium, shape = alpha, rate = beta)
  all_psi2[as.character(c0), "actual"] = psi_2
}

############################## e) ###################################
######## A: Method of rounding ####################

# Create a function for g*, the discretisation of g
g_discrete <- function(x, h, m, shape, rate){
  lower_bound <- max(0, x - h/2)
  upper_bound <- min(h*m, x + h/2)
  if ((((x/h) %% 1) != 0)|
    (x < 0) | (x > h*m)){
    mass <- 0
  } else if (lower_bound == 0){
    mass <- pgamma(upper_bound, shape = shape, rate = rate)
  } else if (upper_bound == h*m){
    mass <- 1 - pgamma(lower_bound, shape = shape, rate - rate)
  } else {
    mass <- pgamma(upper_bound, shape = shape, rate = rate) -
      pgamma(lower_bound, shape = shape, rate = rate)
  }
  return(mass)
}

# Create a function for G*, the discretisation of G
```

```
G_discrete <- function (x, h, m, shape, rate){
  # Find corresponding discrete level, nearest multiple of h less than x
  x_discrete <- floor(x/h)*h
  return(pgamma(x_discrete + h/2, shape = shape, rate = rate))
}


for (h in c(1, 5)){
  for (m in c(150, 300)/h){
    for (init_capital in v_c0){
      c0 <- init_capital
      psi_1 <- function(c){
        return(1 - G_discrete(c + premium, h, m, shape = alpha, rate = beta))
      }
      sum_index <- 0:floor(c0 + premium)
      psi_2 <- 0
      for (i in sum_index){
        psi_2 <- psi_2 + g_discrete(i, h, m, shape = alpha, rate = beta) *
          psi_1(c0 + premium - i)
      }
      psi_2 <- psi_2 + psi_1(c0)
      all_psi2[as.character(c0), paste("h", h, "m", m, sep = "")] <- psi_2
    }

  }
}

######## B: Method of bounds   ####################

# Use discretize function in actuar
upper_lim <- 1000
for (c0 in v_c0){
  for (method in c("lower", "upper")){
    for (d in c(1, 5)){
      discretised_gamma <- discretise(pgamma(x, shape = alpha, rate = beta),
                                      from = 0, to = upper_lim,
                                      step = d,
                                      method = method)
      discrete_gamma_pmf <- function(x){
        if (((x/d)%%1) != 0){
          mass <- 0
        } else if ((x < 0)|(x/d + 1 > length(discretised_gamma))){
          mass <- 0
        } else {
          mass <- discretised_gamma[x/d + 1]
        }
        return(mass)
      }

      discrete_gamma_cdf <- function(x){
        index <- floor(x/d + 1)
        if (index > length(discretised_gamma)){
          cdf <- 1
        } else if (index < 1){
          cdf <- 0
        } else{
          cdf <- sum(discretised_gamma[1:index])
        }
        return(cdf)
      }

      psi_1 <- function(c){
        1 - discrete_gamma_cdf(c + premium)
      }

      sum_index <- 0:floor(c0 + premium)
      psi_2 <- 0
      for (i in sum_index){
        psi_2 <- psi_2 + discrete_gamma_pmf(i) * psi_1(c0 + premium - i)
      }
      psi_2 <- psi_2 + psi_1(c0)
      all_psi2[as.character(c0), paste(method, "_d", d, sep ="")] <- psi_2
    }
  }
}

# Plot differences
all_psi2_long <- melt(all_psi2, id = c("c0"),
                      variable.name = "Method",
                      value.name = "psi_2")

plot_approx_methods <- ggplot() + geom_line(data = all_psi2_long %>%
                                              filter(Method != "actual"),
                     aes(x = c0, y = psi_2, col = Method)) +
  geom_line(data = all_psi2_long %>% filter(Method == "actual"),
            aes(x = c0, y = psi_2), linetype = "dotted",
            size = 1.1) + theme_bw() +
  labs(title = "Comparison␣of␣approximation␣methods",
        x = expression(c[0]),
       y = expression(psi[2](c[0])))

if (export == TRUE){
  ggsave(paste(report_path, "Approximations.pdf", sep = ""), plot_approx_methods)
}
```

B.4 Task 3

```
library(actuar)
library(dplyr)
library(ggplot2)
library(gridExtra)

# Clear everything to prevent unintentional reuse of objects
```

```
rm(list=ls())

# Toggle to export plots
report_path <-
  "C:/Users/johnn/Dropbox/Uni/2017␣S2/ACTL3162/Assignment/Report/images/"
export <- TRUE

# Points to evaluate CDF at
x <- 10*(1:5)

# Gamma distribution parameter
alpha <- 5
beta <- 0.8

# Binomial distribution parameters
n <- 5
p <- 0.997
q <- 1 - p

# Mass dispersal approximation parameters
h <- 1
m <- 70

################# Part A ################
round(pgamma(x, shape = n*alpha, rate = beta),7)

# Panjer values a and b
a <- -p/(1-p)
b <- (n+1)*p/(1-p)

# Discretisation of gamma
gamma_discrete <- discretise(pgamma(x, shape = alpha, rate = beta),
                             from = 0,
                             to = h*m,
                             step = h,
                             method = "rounding")

# Function to output value of discretisation
f_gamma_discrete <- function(x){
    mass <- gamma_discrete[x + 1]
  return(mass)
}

# Initial values
g0 <- f_gamma_discrete(0)
f0 <- (q + p * g0)^n

f_y1 <- aggregateDist(method = "recursive", model.freq = "binomial",
                      model.sev = gamma_discrete,
                      size = 5, prob = 0.99, convolve = 0, x.scale = 1)
```

```
# Run through panjer recursion
f_y <- f0
for (s in 1:max(x)){
  j <- 1:s
  f_y[s + 1] <- 1/(1 - a * g0) * sum((a + b * j / s) *
                                 f_gamma_discrete(j) * f_y[s - j + 1])
}

# Store values as dataframe
panjer_df <- data.frame(y = 0:(length(f_y)-1), f = f_y)
for (row in 1:nrow(panjer_df)){
  panjer_df$F_y[row] <- sum(panjer_df$f[1:row])
}
panjer_df <- panjer_df %>%
  mutate(True_density = dgamma(y, shape = n*alpha, rate = beta),
         True_distribution = pgamma(y, shape = n*alpha, rate = beta))

panjer_plot_f <- ggplot(data = panjer_df, aes(y, f_y)) +
  geom_bar(stat = ’identity’) +
  labs(title = "Panjer’s␣recursion␣mass␣function", y = expression(f[Y](y))) +
  theme_bw() + geom_line(aes(x = y, y = True_density), size = 0.7, color = ’red’,
                         alpha= 0.7)
panjer_plot_f

interleaved_Fy <- rep(0, 2*nrow(panjer_df))
interleaved_Fy[seq(from = 1, by = 2, length.out = nrow(panjer_df))] <- panjer_df$F_y
interleaved_Fy[seq(from = 2, by = 2, length.out = nrow(panjer_df))] <- panjer_df$F_y

interleaved_y <- interleaved_Fy
interleaved_y[seq(from = 1, by = 2, length.out = nrow(panjer_df))] <- panjer_df$y
interleaved_y[seq(from = 2, by = 2, length.out = nrow(panjer_df))] <-
  c(panjer_df$y[-1], tail(panjer_df$y, 1) + 0.1)

panjer_df_F <- data.frame(y = interleaved_y, F_y = interleaved_Fy)

panjer_plot_F <- ggplot(data = panjer_df_F, aes(y, F_y))  + geom_line() +
  labs(title = "Panjer’s␣recursion␣distribution␣function",
       y = expression(F[Y](y))) +
  theme_bw() +
  geom_line(data = panjer_df, aes(x = y, y = True_distribution), color = ’red’,
            size = 0.7, alpha = 0.8)
panjer_plot_F

panjer_combined <- grid.arrange(panjer_plot_f, panjer_plot_F, ncol = 2)

panjer_comparison <- panjer_df %>%
  filter(y %in% x) %>%
  select(-f, -True_density) %>%
  mutate(Difference = F_y - True_distribution) %>%
  mutate(perc_diff = round(100*Difference/True_distribution, 3)) %>%
  round(6)
```

```
# Export graph
if (export == TRUE){
  ggsave(paste(report_path, "Panjer_combined.pdf", sep = ""), panjer_combined,
         width = 10, height = 5)
}
```