Delft University of Technology
Faculty Electrical Engineering, Mathematics and Computer Science
Delft Institute of Applied Mathematics

Developing an R-package for the Gamma distribution

(Ontwikkelen van een R-package voor de Gamma verdeling)

A thesis submitted to the
Delft Institute of Applied Mathematics
in partial fulfillment of the requirements

for the degree

BACHELOR OF SCIENCE
in
APPLIED MATHEMATICS

by

Kilian Buis

Delft, the Netherlands
9 July 2020

Thesis Committee:
Dr. P. Chen (supervisor)
Dr. J.G. Spandaw

Abstract

Since the gamma distribution is one of the most important models, and no convenient statistical tools for this distribution are available, the aim of this project is to construct an R package for the gamma distribution. In this package five functions are created, that can be used to either generate gamma distributed random variables, to estimate the parameters of a gamma distribution and to construct statistical limits for the gamma distribution. In this report, the methods to do so are explained, implemented in R, and the performance of the methods have been tested. Lastly, an example is given to illustrate the application of the package.

Contents

1 Introduction

In the field of statistics and probability, the gamma distribution is one of the most important parametric models. Its importance is largely due to its relation to exponential and normal distributions. Because of the skewness of the gamma distribution, it is a very useful model in a lot of different areas when the normal distribution is not a convenient model [2]. For example, many problems in telecommunication networks can be modelled by queuing systems and waiting times, for which a gamma distribution can be used. For an application in this area, see Section 2.3. Moreover, the gamma distribution can be used to monitor the amount of daily rainfall [2]. Also, it is found to be an appropriate model for the distance between mutations on a DNA strand. Lastly, the gamma distribution offers a good fit to many lifetime data sets [3].

These are only a few examples, but all these examples show the importance of the gamma distribution. Because the gamma distribution is such an important distribution, it might be relevant to construct statistical tools. Although there have been many studies that investigated the properties of the gamma distribution, there is still a lack of statistical tools to facilitate practical applications. To fill this gap, the aim of this project is to develop an R package for the gamma distribution. The R package I will construct contains five functions that should be able to conduct the following tasks: generate gamma distributed random variables; estimate the model parameters; and construct statistical limits for a gamma distribution.

The methods and algorithms I use to implement these functions, I have not invented myself. These have already been developed and I am putting these methods together in one R package, that can be used for data that is gamma distributed. The article *"Logarithmic Transformation-Based Gamma Random Number Generators"* [1], written by Bowei Xi, Kean Ming Tan and Chuanhai Liu, proposes methods to generate gamma distributed random variables. In the article *"Closed-Form Estimators for the Gamma Distribution Derived From Likelihood Equations"* [2], Zhi-Sheng Ye and Nan Chen introduce a method to obtain closed form estimators of the gamma distribution. Lastly, in the article *"Approximate Statistical Limits for a Gamma Distribution"* [3], Piao Chen and Zhi-Sheng Ye state 3 methods to construct confidence limits, prediction limits and tolerance limits, respectively. To construct tolerance intervals, I use the method described in the article *"Normal-Based Methods for a Gamma Distribution: Prediction and Tolerance Intervals and Stress-Strength Reliability"* [4] by K. Krishnamoorthy, Thomas Mathew and Shubhabrata Mukherjee.

The remainder of this report is organized as follows. Section 2 gives an introduction to the gamma distribution. In this section, the properties of a gamma distribution are mentioned, and an application of the gamma distribution is illustrated. In Section 3, I will tell about the already existing methods to generate gamma distributed random variables, why these methods are not satisfactory, and illustrate the newly proposed method to generate gamma distributed random variables. Also, the performance is checked by calculating the acceptance rate. Section 4 is about estimating the model parameters of a gamma distribution and is organized in the same way. First the already existing methods are mentioned, and then the new method is introduced. This new method is also compared with the method of moments. Section 5 is about constructing statistical limits. Also in this section the already existing methods are mentioned, and the new methods and algorithms are introduced. Furthermore, simulations are conducted to asses the performance of the new methods. In addition, the method to construct confidence intervals is compared with the bootstrap method. In Section 6, I give an example to show how the package can be used for a gamma distribution. Finally, Sections 7 and 8 conclude and discuss the report.

2 Gamma distribution

The gamma distribution, denoted as $\text{gam}(k, \theta)$ is a two-parameter distribution that has three common parametrizations in use. Throughout this report, I will use the one with probability density function (PDF)

$$f_{gam}(x) = \frac{\theta^k}{\Gamma(k)} x^{k-1} e^{-\theta x}, x > 0, \tag{1}$$

where k is the shape parameter and θ is the rate parameter [3]. Note, that the scale parameter, which I will sometimes refer to as β, and that is used in another parametrization of the gamma distribution, is given by the inverse of the rate parameter. Furthermore, $\Gamma(\cdot)$ is the gamma function. For complex numbers with a positive real part the gamma function is defined via a convergent improper integral

$$\Gamma(z) = \int_0^\infty x^{z-1} e^{-x} dz,$$

with $\Gamma(1) = 1$ and $\Gamma(z+1) = z\Gamma(z)$. Also, for all $n \in \mathbb{N}$, it holds that $\Gamma(n+1) = n!$.

2.1 The parameters of the gamma distribution

As said before, k is the shape parameter, while θ is referred to as the rate parameter. A gamma distribution with $\theta = 1$ is known as a standard gamma distribution. Note that
$X \sim \text{gam}(k, \theta) \iff \theta X \sim \text{gam}(k, 1)$. Hence, it is easy to transform a non-standard gamma distribution into a standard gamma distribution, and vice versa. As the name suggests, the shape parameter k controls the shape of the distribution. To be precise, the distribution could take 3 different shapes, depending on the shape parameter k [5].

Firstly, when $k < 1$, the gamma distribution is exponentially shaped and is asymptotic to both the vertical and horizontal axis. In Figure 1 I visualize this by showing the probability density function of a random variable that is $\text{gam}(\frac{1}{2}, 1)$ distributed.

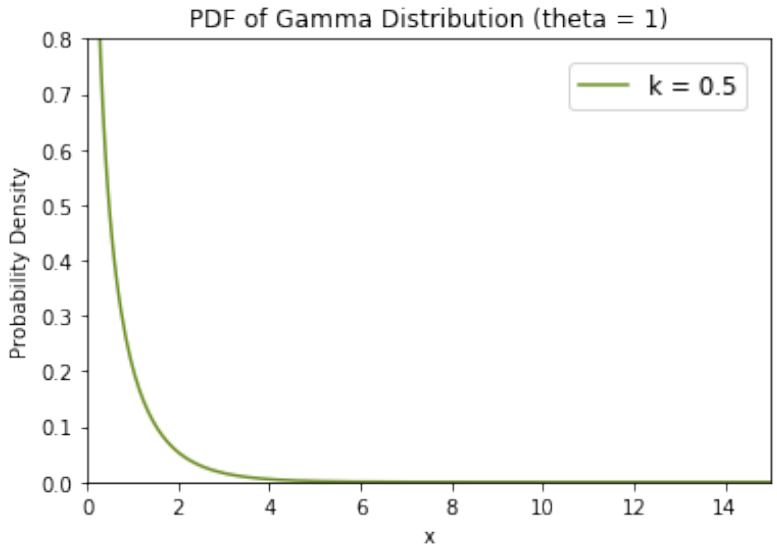


Figure 1: The probability density function of a $\text{gam}(\frac{1}{2}, 1)$ distributed random variable. It can be seen that the PDF is exponentially shaped and asymptotic to both the vertical and horizontal axis.

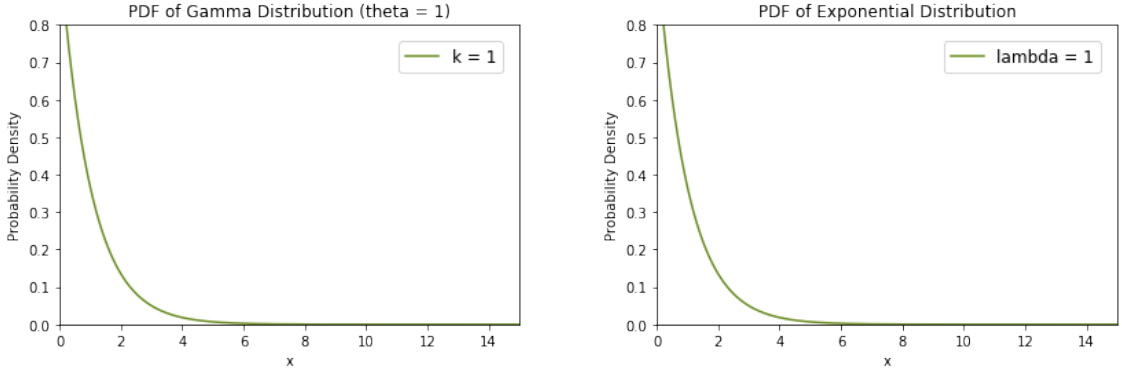
The second case when $k = 1$, gives an exponential distribution with expectation $\frac{1}{\theta}$. This can be verified by substituting $k = 1$ in (1). Doing so gives the probability density function

$$f_{gam}(x) = \theta e^{-\theta x}.$$

The probability density function of an exponentially distributed exp(λ) random variable is given by

$$f_{exp}(x) = \lambda e^{-\lambda x},$$

and has expectation equal to $\frac{1}{\lambda}$. Now it is easy to see, that if one substitutes $\theta = \lambda$, the two probability density functions above are equal. Hence, a gamma distribution with shape parameter $k = 1$ is equal to an exponential distribution with expectation $\frac{1}{\theta}$. In Figure 2 I visualize this by showing both the probability density function of a gam(1,1) and an exp(1) distributed random variable that. Note that for the gamma distribution I used $\theta = 1$, which meant that for the exponential distribution, I had to use $\lambda = 1$. As expected, the two probability density functions are equal.



(a) The probability density function of a gam$(1, 1)$ distributed random variable.

(b) The probability density function of an exp(1) distributed random variable.

Figure 2: The probability density functions of a gam(1,1) and a exp(1) distributed random variable. It can be seen that both functions are similar.

The last case is the case when $k > 1$. Then the gamma distribution assumes a unimodal, but skewed shape. I visualize this by showing the probability density function of a random variable that is gam$(5, 1)$ distributed. See Figure 3. As can be seen, the probability density function is slightly right skewed. Note that the skewness reduces as the value of k increases. This can also be seen in Figure 3, where I show to probability density function of a random variable that is gam$(20, 1)$ distributed.

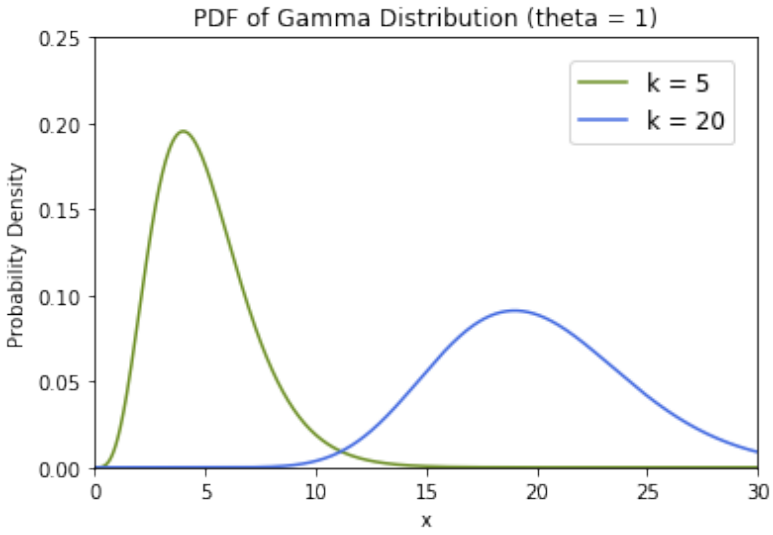


Figure 3: The probability density functions of a gam$(5, 1)$ distributed random variable and a gam$(20, 1)$ distributed random variable. It can be seen that a gamma distribution with a higher shape parameter makes the PDF look more symmetric.

The fact that the skewness reduces as the value of k increases, can be explained using the following statement:

"If X is a gam(k, θ) distributed random variable and Y is a normal random variable with the same mean and variance as X, then $F(x) \approx F(y)$ if the shape parameter k is large relative to the scale parameter $\beta = \frac{1}{\theta}$ [6]. "

The expectation and variance of a random variable X that is gam(k, θ) distributed are given by

$$\mathbb{E}(X) = \frac{k}{\theta}$$

and

$$\text{Var}(X) = \frac{k}{\theta^2}.$$

So the statement above can be rephrased as follows: "If X is gamma distributed and the shape parameter k is large relative to the scale parameter $\beta = \frac{1}{\theta}$, then X is approximately a normal random variable with mean $\frac{k}{\theta}$ and variance $\frac{k}{\theta^2}$, i.e. $X \sim \mathcal{N}(\frac{k}{\theta}, \frac{k}{\theta^2})$."

Hence, if the value of k increases, k will become large relative to $\frac{1}{\theta}$, and the probability density function will look much more like a normal distribution. Since all normal distributions are symmetric, the skewness of the gamma distribution is reduced when the value of k increases.

This can also be explained by looking at the skewness. For a gam(k, θ) distribution, the skewness is given by $\frac{2}{\sqrt{k}}$. Hence, when k gets large, the skewness tends to zero, meaning the distribution will become more symmetric.

2.2 Properties of the gamma distribution

In Section 2.1 I already mentioned some of the properties of a gamma distribution. In this section, I will list some more properties. Most of these properties are relations between the gamma distribution and another distribution. Because of these relations, it emphasizes once again how important the gamma distribution is.

- For $i = 1, ..., n$, if $X_i \sim \text{gam}(k_i, \theta)$, then $\sum_{i=1}^{n} X_i \sim \text{gam}\left(\sum_{i=1}^{n} k_i, \theta\right)$.
- If $X \sim \text{gam}(k, \theta)$, then for any positive scalar c, it holds that $cX \sim \text{gam}\left(k, \frac{\theta}{c}\right)$.
- If $X_1, ..., X_n$ are n independent and identically distributed random variables that follow a $\exp(\lambda)$ distribution, then $\sum_{i=1}^{n} X_i \sim \text{gam}(n, \lambda)$ [7].
- If $X \sim \text{gam}(\frac{v}{2}, \frac{1}{2})$, then X is equal to a chi-squared distribution with v degrees of freedom, i.e. $X \sim \chi^2(v)$ [8].
- If $X \sim \chi^2(v)$, then for any positive scalar c, it holds that $cX \sim \text{gam}(\frac{v}{2}, \frac{1}{2c})$.
- If $X \sim (k_1, \theta)$ and $Y \sim \text{gam}(k_2, \theta)$ and X and Y are independent, then $X/(X + Y)$ has a Beta distribution with parameters k_1 and k_2, i.e. $X/(X + Y) \sim \text{Beta}(k_1, k_2)$ [8].
- If $Z \sim \mathcal{N}(0, 1)$, then $Z^2 \sim \text{gam}(\frac{1}{2}, \frac{1}{2})$ [8].
- If k is an integer, the gamma distribution is equal to the Erlang distribution.

2.3 Application of the gamma distribution

In Section 2.2 I gave a few properties of the gamma distribution. In this section I will give an application of the gamma distribution, introduced by Aerin Kim [9]. The application is based on the last property I mentioned in Section 2.2. As I said, if k is an integer, then the gamma distribution is equal to the Erlang distribution. The Erlang distribution can be used to predict waiting times. For example, the time it takes before the kth call in a call center comes in. These calls (events) can occur independently with some event rate and are modelled with a Poisson process.

In practice, when using the gamma/Erlang distribution for predicting waiting times, the shape parameter k will be the number of events for which one is waiting, and the rate parameter θ will be the rate of the happening of these events. T will be the random variable that is used for the waiting time until the kth event. With this notation, if one fixes the rate θ, it is expected that if the number of events (k) increases, the waiting time will be longer. This is indeed true, as can be seen in Figure 4. In this example, the rate was fixed as $\theta = 1$ and the number of events was taken as either 1, 4 or 8. As expected, it can be seen that the average waiting time will be higher, as the number of events increases.

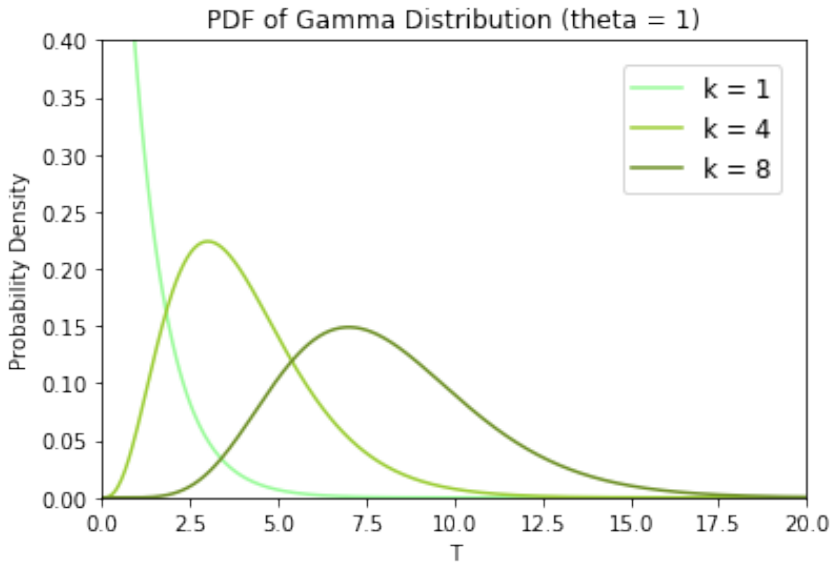


Figure 4: Three distributions of the waiting times. The event rate is set at 1, while the number of events is taken as 1, 4 or 8.

On the other hand, if one fixes the number of events k, it is expected that the waiting will be shorter when the event rate θ is increased. This is verified with the use of Figure 5. In this example, the number of events was fixed as $k = 10$. The rate was taken as either 1, 3 or 5. As expected, the average waiting time is shorter if the the event rate increases.

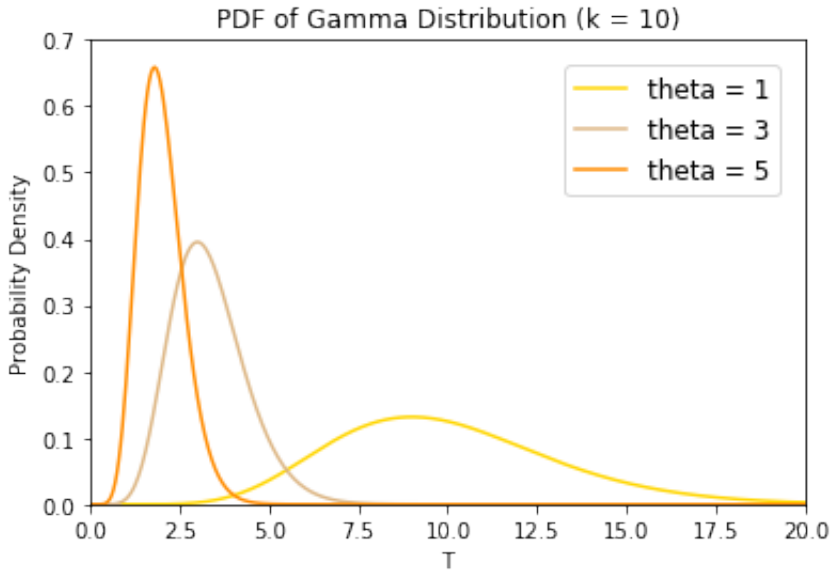


Figure 5: Three distributions of the waiting times. The number of events is set at 10, while the event rate is taken as 1, 3 or 5.

This simple example is only one of the applications where the gamma distribution is being used, but it gives a clear signal that the gamma distribution is an important distribution, and it underlines the fact that statistical tools need to be constructed.

3 Generate gamma distributed random variables

3.1 Introduction

Developing an efficient method to generate gamma distributed random variables is important for Monte Carlo methods. In the past, many methods have already been developed. However, these methods are often only sufficient for a limited range of shape parameter values, either for $k < 1$ or $k \geq 1$ [1]. For example, Ahrens and Dieter (1974) [10] found out that gamma distributed random variables with a shape parameter $k \geq 1$ can be generated using a combination of two other generators, one generator for shape parameters that are integers and another generator for $k < 1$. This methods seems to work fine. However, in this algorithm, they need to generate two random numbers, and use these two random numbers in order to get the one desired random number. This can take a long time, and thus is unsatisfactory.

Because of the fact that the current methods do not always work sufficient, Bowei Xi, Kean Ming Tan and Chuanhai Liu, have proposed two new methods to generate gamma distributed random variables [1]. One algorithm works for all shape parameters $k > 0$, and one is constructed only for shape parameters $k \leq 1$. Since the already existing method that is used in R works fine for shape parameters $k > 1$, I will use that existing method in my package to generate gamma distributed random variables with shape parameter $k > 1$. However, for shape parameters $k \leq 1$, the already existing method in R uses a similar method as the method of Ahrens and Dieter (1974) [10], as explained above. Since that method is not always sufficient, I will use the algorithm proposed by Bowei Xi, Kean Ming Tan and Chuanhai Liu, to generate gamma distributed random variables with a shape parameters $k \leq 1$.

3.2 Main idea of the method

As already said above, I will only use the proposed algorithm for shape parameters $k \leq 1$. In this section, I will state the main idea of this algorithm. To do that, I will first explain the idea behind the ratio-of-uniforms (ROU) method.

The ROU method is a special case of a acceptance-rejection sampling method. It starts by assuming that there is a non-negative function $h(t)$ with finite integral M_h. In this way, the function $h(t)/M_h$ has an integral equal to 1 and hence is a probability density function. Then, if the random point (U, V) is uniformly distributed over what I now refer to as the ROU region

$$\mathcal{C} = \{(u, v) : 0 \leq u \leq \sqrt{h(v/u)}\}$$

then $T = V/U$ has density $h(t)/M_h$ [1]. In other words, to generate random numbers from the distribution with density $h(t)/M_h$, a random point (U, V) needs to be generated uniformly over the ROU Region $\mathcal{C}$. The ratio $T = V/U$ can be used as the random number.

To do this, upper bounds of both u and v are determined to construct a rectangle that covers the ROU region. This rectangle, which I call $\mathcal{C}^*$, is given by

$$\mathcal{C}^* = \{(u, v) : 0 \leq u \leq \sqrt{h(t)}, v_{min} \leq v = tu \leq v_{max}\}.$$

Now the point (U, V) is uniformly drawn over this rectangle. The point is only accepted, if the point falls inside the ROU-region. To illustrate the idea, I use an example, where random numbers are generated that are gamma distributed [1]. In Figure 6, the grey area is the ROU-region, while the blue rectangle is the rectangle that covers the ROU-region. In this rectangle, the point (U, V) is uniformly drawn. If the point happens to be $(U, V) = (0.8, -2.0)$, the point falls outside the ROU region, and the point will not be accepted. On the other hand, if the drawn point is $(U, V) = (0.4, -1.0)$, the point falls inside the ROU region, and the point will be accepted. The ratio $T = U/V$ will be used as random number.

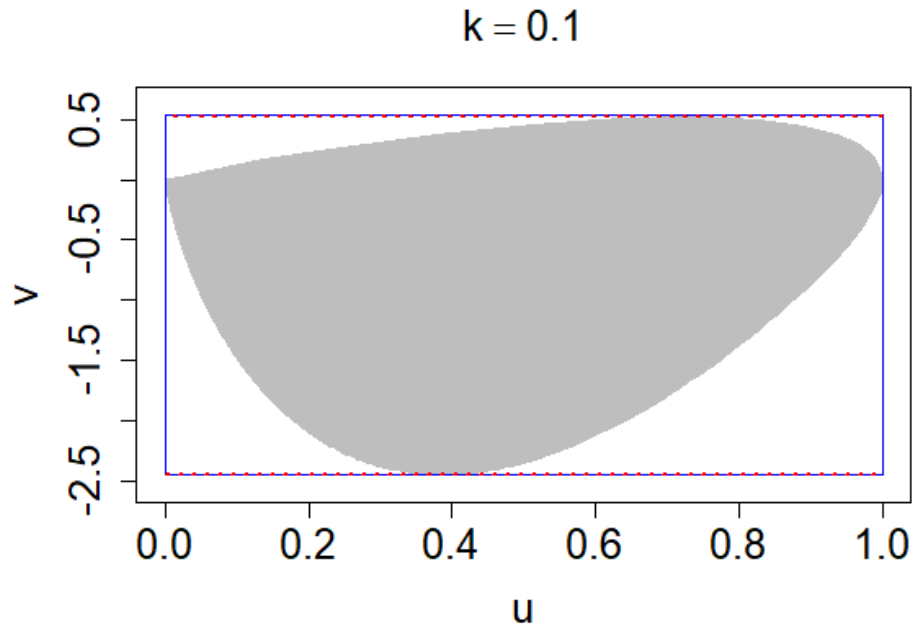


Figure 6: The ROU region is given by the grey area. The blue rectangle covers the ROU-region. The vertical blue lines show the range $0 \leq u \leq \sqrt{h(t)}$. The horizontal blue lines show the range $v_{min} \leq v \leq v_{max}$. From "Logarithmic Transformation-Based Gamma Random Number Generators ," by B. Xi, K.M. Tan, C. Liu, 2013 [1].

If one now want to use this method to generate gamma distributed random variables with a shape parameter $k \leq 1$, first a transformation T is needed. Then the ROU-method is used on this transformed random variable T. Finally, the inverse transformation can be used to obtain a random number that is gamma distributed.

The transformation that is considered is a logarithmic transformation:

$$T = k \ln(X)$$

or, equivalently $X = \exp(T/k)$. Following the ROU method, let

$$h(t) = \exp(t - e^{t/k}).$$

The ROU region is then given by

$$\mathcal{C} = \{(u, v) : 0 \leq u \leq \sqrt{h(t)}, -\infty < t = \frac{v}{u} < \infty\}.$$

The rectangle I use to cover the ROU region is given by

$$\mathcal{C}^* = \{(u, v) : 0 \leq u \leq \sqrt{h(t)}, v_{min} < v = tu < v_{max}\}.$$

Since $\max\sqrt{h(t)} = \left(\frac{k}{e}\right)^{k/2}$, it holds that

$$0 \leq u \leq \left(\frac{k}{e}\right)^{k/2}. \tag{2}$$

Similarly, for $v = tu$, it can be shown that

$$-\frac{2}{e} \leq v \leq \frac{2k}{e(e-k)}. \tag{3}$$

Equations 2 and 3 define the rectangle $\mathcal{C}^*$ that covers the ROU Region $\mathcal{C}$ [1]. Now using the following pseudo-algorithm, gamma distributed random variables can be generated.

Pseudo algorithm to generate n **gamma random numbers given** $k \leq 1$
umax $= (k/e)^{k/2}$, vmin $= -2/e$, vmax $= 2k/(e(e-k))$
$k = 1$
while $k \leq n$ **do**
    u = umax $\times$ Uniform(0,1)
    v = (vmax - vmin) $\times$ Uniform(0,1) + vmin
    t = v/u
    $x = e^{t/k}$
    if $2\ln(u) \leq t - x$ **then**
        Deliver x
        $k = k + 1$

The algorithm starts by first defining the minimum and maximum values of u and v, to define the rectangle $\mathcal{C}^*$ that covers the ROU Region. Then the algorithm does a certain amount of iterations, until a random variable consisting of n random numbers is generated. In each iteration, the random numbers U, V and $T = V/U$ are set up. Then T is transformed by the transformation that is mentioned earlier: $X = \exp(T/k)$. The number X will be returned as random number. Note that the algorithm only returns X when $2\ln(u) \leq t - x$. This is an essential step of the algorithm. Namely, as I explained before, only the points that fall inside the ROU-region are accepted. The line $2\ln(u) \leq t - x$ checks if the values are in the ROU region, and returns them if they do.

3.3 Implementation

In the function I created in my R package to generate gamma distributed random variables, the user can enter the shape parameter. I distinguish between two cases; a shape parameter > 1 and a shape parameter ≤ 1. As said before in Section 3.1, the already existing algorithm for shape parameters $k > 1$ works fine. That is why, in my code I used the already existing function *rgamma* when a shape parameter > 1 is entered.

When a shape parameter ≤ 1 is entered, I generate gamma random numbers using the algorithm described in Section 3.2. To be precise, that algorithm generates random numbers of a standard (unit) gamma distribution. That is a gamma distribution, where the rate parameter are equal to 1. Since the algorithm also needs to work for other values of the rate (and scale) parameter, I divide all my n numbers by the rate parameter θ in the last step of my implementation. This has to do with the fact that $X \sim \text{gam}(k, \theta) \iff \theta X \sim \text{gam}(k, 1)$. Also, this follows because the expectation of a random variable X that is $\text{gam}(k, \theta)$ distributed is equal to $\frac{k}{\theta}$.

One last note; In my code I also included the scale parameter. That means that the same numbers will be generated, no matter if the rate or scale parameter is entered. If no scale or rate parameter is entered, the function uses a default value of 1.

3.4 Validation

In this section, I check my random number generator in terms of acceptance rate. To explain how the acceptance rate can be calculated, I give the following example. Suppose one wants to generate a gamma distributed random variable, consisting of 100 random numbers. With the constructed function in R, these 100 random numbers can be generated. However, in practice this algorithm generates more than 100 numbers, because some of the generated numbers do not fall in the ROU region, and hence will be rejected. Let's say the algorithm needed to generate 140 numbers, in order to get 100 accepted numbers. In this case the acceptance rate is equal to $100/140 = 0.714$. In general the acceptance rate is given by the formula

$$\text{Acceptance rate} = \frac{\text{Amount of numbers that one wants to generate}}{\text{Amount of numbers that are generated by the algorithm}}.$$

To test the algorithm to construct gamma distributed random variables with a shape parameter $k \leq 1$, I calculated for 1000 different shape parameters ($k = 0.001, 0.002, ..., 0.999, 1.000$) the acceptance rate. I did this by generating a sample of 10.000 random numbers that are gamma distributed with one of the above shape parameters, while θ was equal to 1. I then counted for each shape parameter how many iterations the algorithm needed in order to get 10000 random numbers. In this way, I determined for each shape parameter the acceptance rate. In Figure 7, I show these acceptance rates for different values of k with the blue points.

I also determined the theoretical acceptance rate by the formula

$$\text{Theoretical acceptance rate} = \frac{\text{Size of the ROU Region } \mathcal{C}}{\text{Size of the rectangle } \mathcal{C}^*}.$$

Depending on the shape parameter k, the size of the ROU region $\mathcal{C}$ is given by

$$|\mathcal{C}| = \frac{\Gamma(k)k}{2}[1],$$

while the size of the rectangle $\mathcal{C}^*$ that covers the ROU region is given by

$$\text{(umax - umin)} \cdot \text{(vmax - vmin)} = \left(\frac{k}{e}\right)^{k/2} \left(\frac{2k}{e(e-k)} + \frac{2}{e}\right)$$

For every shape parameter I calculated the theoretical acceptance rate and I show these acceptance rates in Figure 7 by the black points.

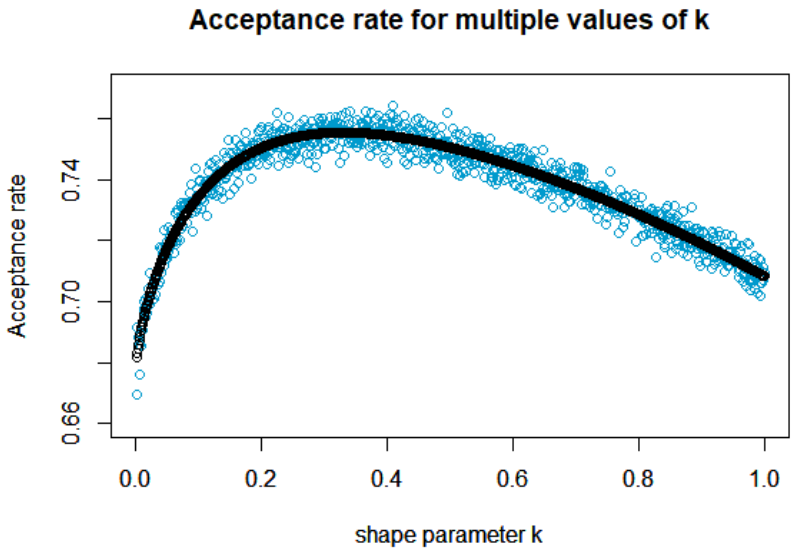


Figure 7: The blue dots indicate the acceptance rates of the algorithm for different values of k. The black dots indicate the theoretical acceptance rates for different values of k.

What can be observed from Figure 7 is that the acceptance rate of the algorithm is very close to the theoretical acceptance rate for every shape parameter $k \leq 1$. From that I may conclude that I implemented the algorithm correctly in my R package. Furthermore, it can be seen that all acceptance rates are higher than 0.660, with even a peak around 0.750. These acceptance rates are pretty high, meaning that the algorithm is fairly efficient.

4 Estimate the model parameters of a gamma distribution

4.1 Introduction

Parameter estimation is very important and can be done in many different ways. The most popular parameter estimation method is the maximum likelihood method. Unfortunately, for the the two-parameter gamma distribution, defined before as (1), the maximum likelihood estimators have no closed-form expressions. These estimators can still be found in another way, but this can be difficult when k is small. Also, it can take a long time.

Another popular parameter estimation method is the method of moments. However, these estimators are not efficient under either small or large samples. Because of the inefficiency of both the maximum likelihood estimators and the method of moments estimators, Zhi-Sheng Ye and Nan Chen have proposed a new method to generate efficient estimators of the parameters of a gamma distribution. [2]. This method I will use in my R package.

4.2 Main idea of the method

The new method to estimate the gamma parameters makes use of the generalized gamma distribution, denotes as $\text{gg}(k, \theta, \gamma)$, where $\gamma > 0$ is the power parameter. It has probability density function

$$f_{gg}(x) = \frac{\gamma \theta^{k\gamma}}{\Gamma(k)} x^{k\gamma - 1} e^{-(\theta x)^{\gamma}}, x > 0.$$

One of the properties of the generalized gamma distribution, is that if $X \sim \text{gam}(k, \theta)$, then $X^{\gamma} \sim \text{gg}(k, \theta, \gamma)$.

Now let $X \sim \text{gam}(k, \theta)$ and $X_1, X_2, ..., X_n$ be n independent identically distributed (iid) copies of X. Using the property above, it is easy to see that $X \sim \text{gg}(\alpha, \beta, \gamma)$ with $\gamma = 1$. This observation is important and suggests the following procedure.

Assume that X follows the above generalized gamma distribution with unknown power parameter γ. Determine with the maximum likelihood method the maximum likelihood estimators of X. This will give 3 estimators; one for k, θ and γ, respectively. I will denote them by k^*, θ^* and γ^*. Lastly, using the fact that a generalized gamma distribution with power parameter $\gamma = 1$ corresponds to an original $\text{gam}(k, \theta)$ distribution, the maximum likelihood estimators of the gamma distribution can be found by substituting $\gamma = 1$ in the (now closed-form) expressions of the estimators k^* and θ^*.

The method above gives the following two estimators for k and θ, denoted by $\hat{k}$ and $\hat{\theta}$, respectively.

$$\hat{k} = \frac{n \sum_{i=1}^{n} X_i}{n \sum_{i=1}^{n} X_i \log(X_i) - \sum_{i=1}^{n} \log(X_i) \sum_{i=1}^{n} X_i}, \tag{4}$$

$$\hat{\theta} = \frac{n^2}{n \sum_{i=1}^{n} X_i \log(X_i) - \sum_{i=1}^{n} \log(X_i) \sum_{i=1}^{n} X_i}, \tag{5}$$

where n is the number of observation of the data set X.

Because the two estimators are found in the same way as the original maximum likelihood estimators would normally be found, it is expected that the performance of the new estimators should roughly be the same as the performance of the maximum likelihood estimators. Indeed, in large-samples, the estimators $\hat{k}$ and $\hat{\theta}$ are strongly consistent estimators of k and θ, meaning that

$$\lim_{n\to\infty} \hat{k} = k \text{ and } \lim_{n\to\infty} \hat{\theta} = \theta.$$

Also, when $n \to \infty$, the estimators $\hat{k}$ and $\hat{\theta}$ are asymptotically normally distributed with expectation

$$\mathbb{E}[\sqrt{n}(\hat{k} - k, \hat{\theta} - \theta)] \to \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

Lastly, the variance curves of the proposed estimators 4 and 5 are almost the same as the original (non-closed form) maximum likelihood estimators. These three properties suggest that the new estimators 4 and 5 perform well.

However, sometimes, the estimators (4) and (5) can be biased. Unbiased estimators for the shape and rate parameter are not available, but biased-corrected estimators for k and θ do exist. These are given by

$$\tilde{k} = \frac{n-1}{n+2}\hat{k} = \frac{n(n-1)\sum_{i=1}^{n} X_i}{(n+2)[n\sum_{i=1}^{n} X_i \log(X_i) - \sum_{i=1}^{n}\log(X_i)\sum_{i=1}^{n} X_i]} \tag{6}$$

$$\tilde{\theta} = \frac{n-1}{n+2}\hat{\theta} = \frac{n^2(n-1)}{(n+2)[n\sum_{i=1}^{n} X_i \log(X_i) - \sum_{i=1}^{n}\log(X_i)\sum_{i=1}^{n} X_i]}. \tag{7}$$

These two estimators, (6) and (7), are the estimators I use in my code. Note that the scale parameter, denoted by β, can also be of interest. That is why I also mention the closed form estimator of the scale parameter. An unbiased estimator for the scale parameter is given by [2]

$$\tilde{\beta} = \frac{1}{n(n-1)}\left(n\sum_{i=1}^{n} X_i \log(X_i) - \sum_{i=1}^{n}\log(X_i)\sum_{i=1}^{n} X_i\right). \tag{8}$$

This estimator (8) I will also include in my R code.

4.3 Implementation

The function that I created in my R package has as input a gamma distributed data set. The output will be the shape, rate and scale (= 1/rate) parameter that fit the best onto this data set. I do that by returning the bias-corrected estimators for k and θ and by returning the unbiased estimator for the scale parameter.

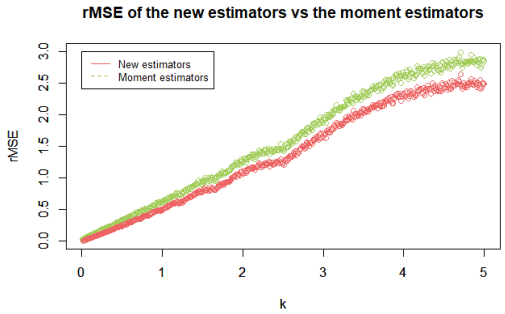
4.4 Validation

As said in Section 4.2, the proposed estimators (4) and (5) are strongly consistent, asymptotically normally distributed, and have almost the same variance curve as the original (non-closed form) maximum likelihood estimators. On top of that, the biases and rMSE's of the proposed estimators is almost the same compared with the original maximum likelihood estimators. The bias-corrected estimators (6), (7) and (8) reduce the bias even more, which improves the performance of the estimators. Taking all this into account, it can be concluded that the proposed closed-form estimators (6), (7) and (8) perform well. Because the proposed estimators do have closed forms, while the maximum likelihood estimators do not, I prefer the proposed estimators over the maximum likelihood estimators.

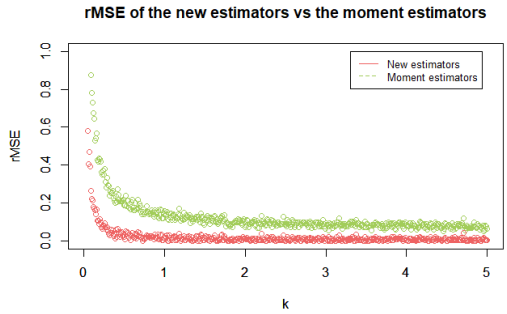
As said before in Section 4.1, the method of moments is also a popular method to estimate model parameters. To check which estimators perform better, I tested both the newly proposed estimators (6 and 7) and the method of moments estimators in terms of root mean squared error (rMSE). The method of moments estimators for the shape and rate parameter are given by

$$k_M = \frac{\left(\sum_{i=1}^n X_i\right)^2}{n\sum_{i=1}^n X_i^2 - \left(\sum_{i=1}^n X_i\right)^2} \text{ and } \theta_M = \frac{n\sum_{i=1}^n X_i}{\sum_{i=1}^n X_i^2 - \left(\sum_{i=1}^n X_i\right)^2}.$$

For multiple values of k, and using a sample size of $n = 50$, I determined the rMSE's for both the shape and rate parameters, using either the newly proposed estimators and the method of moments estimators. See Figure 8.



(a) The rMSE of the shape parameter, using both the new estimators and the method of moments estimators.



(b) The rMSE of the rate parameter, using both the new estimators and the method of moments estimators.

Figure 8: The rMSE of the shape and rate parameter, using both the new estimators and the method of moments estimators. It can be seen that in both cases the rMSE of the new estimators is lower than the rMSE of the moment estimators.

In both cases, it can be concluded that the newly proposed estimators perform better in terms of rMSE than the method of moments estimators. That is why I prefer to use the new estimators in my R package.

5 Construct statistical limits for a gamma distribution

In this section I will treat three methods, proposed by Piao Chen and Zhi-Sheng Ye [3], to construct statistical limits. The three limits that I will construct with these methods are confidence limits, prediction limits and tolerance limits. Doing so, it requires the use of an algorithm to generate realizations of the parameters k and θ. This algorithm is also proposed by Piao Chen en Zhi-Sheng Ye. The idea behind this algorithm is shown in Appendix A.

5.1 Confidence limits

5.1.1 Introduction

Before explaining why confidence limits are of interest, I will first state the definition of a confidence interval, a lower confidence limit and an upper confidence limit.

Definition: Assume n historical data, i.e. $x_1, ..., x_n$ from a gamma distribution with shape parameter $k > 0$, rate parameter $\theta > 0$ and scale parameter $\beta = \frac{1}{\theta} > 0$ are available. Consider two statistics $L = L(x_1, ..., x_n)$ and $U = U(x_1, ..., x_n)$. If

$$P(L \leq k \leq U) = 1 - \alpha,$$

then the interval $[L, U]$ is called an $(1-\alpha)$ confidence interval for the shape parameter k. When U equals ∞, L is called an $(1-\alpha)$ lower confidence limit for the shape parameter k. Similarly, when L equals $-\infty$, U is called an $(1-\alpha)$ upper confidence limit for the shape parameter k. The same arguments hold for the rate parameter θ and the scale parameter β.

In other words, a confidence interval provides an $(1-\alpha)$ confidence of an yet unknown parameter being inside that interval. This is for example very useful, when one wants to know the expectation of a gamma distributed data set with yet unknown parameters.

5.1.2 Main idea of the method

The first step in order to construct confidence limits and intervals is to use Algorithm I, explained in Appendix A. Using this algorithm, B (usually $B = 10000$) realizations of the shape parameter k and the rate parameter θ are constructed. In the same way, also B realizations of the scale parameter β can be constructed.

To now construct a $(1-\alpha)$ lower confidence limit for the shape parameter k, all the B realizations of the shape parameters, called $k_1, ..., k_B$ are put into a list. The values are sorted in ascending order and the 100αth percentile is used as the $(1-\alpha)$ lower confidence limit for the shape parameter k. Similarly, the $100(1-\alpha)$th percentile is used as the $(1-\alpha)$ upper confidence limit for the shape parameter k.

Lastly, to construct a $(1-\alpha)$ confidence interval for k, the interval is taken, between the $(1-\frac{\alpha}{2})$ lower confidence limit and the $(1-\frac{\alpha}{2})$ upper confidence limit.

Note that this method works completely the same for confidence limits and intervals for the rate and scale parameter, by only replacing k with θ or β.

5.1.3 Implementation

In the first part of the function I created in my R package, I set up B realizations of the parameters k, θ and β. That part of the code is explained in Appendix A. After that, confidence limits and confidence intervals are constructed, in the way I explained in Section 5.1.2.

The function I created has as input the parameter *lim*. I did this, so that an user can ask for only a lower confidence limit, only an upper confidence limit or a confidence interval. Another parameter I use in my function is *par*. This parameter I created, so that the user can decide for which parameter (shape, rate or scale), the limits or intervals will be returned. Note, that by default the confidence interval for the shape parameter will be returned.

Lastly, when an user does not input a value for α and/or B, my function will also use a default value. For B that is 10000 and for α that is 0.05, such that a 95% confidence limit/interval will be returned.

5.1.4 Validation

To check the performance of the method and of my function, I performed multiple Monte Carlo simulations. In all simulations, I considered multiple values of k, namely 0.05, 0.1, 0.5, 1 and 1.5. I set $\theta = 1$ and I considered small sample sizes, i. e. $n = 3, 5, 10$.

To check the performance of the lower confidence limit, I performed a Monte Carlo simulation, where in 10.000 replications, I counted how many times the lower confidence limit for k was indeed lower than the actual value of k. This number I divided by 10.000, to get a number between 0 and 1; the coverage probability. The coverage probabilities of the lower confidence limit can be seen in Table 1.

n	**k = 0.05**	**k = 0.1**	**k = 0.5**	**k = 1**	**k = 1.5**
3	0.979 / 0.950	0.981 / 0.952	0.965 / 0.927	0.959 / 0.921	0.957 / 0.909
5	0.979 / 0.940	0.980 / 0.944	0.971 / 0.931	0.961 / 0.918	0.960 / 0.917
10	0.972 / 0.922	0.977 / 0.939	0.971 / 0.937	0.963 / 0.923	0.963 / 0.915

Table 1: Coverage probabilities of the lower confidence limit. For each combination of k and n, the nominal coverage probabilities ($= 1 - \alpha$) are 95% (left) and 90% (right).

To check for the upper confidence limit I did exactly the same, only now I counted how many times the upper confidence limit for k was indeed larger than the actual value of k. Again, I divided that number by 10.000, to get the coverage probabilities. The results are stated in Table 2.

n	**k = 0.05**	**k = 0.1**	**k = 0.5**	**k = 1**	**k = 1.5**
3	0.955 / 0.907	0.960 / 0.911	0.965 / 0.923	0.970 / 0.924	0.965 / 0.924
5	0.952 / 0.910	0.957 / 0.917	0.968 / 0.927	0.966 / 0.926	0.967 / 0.924
10	0.959 / 0.914	0.962 / 0.913	0.969 / 0.924	0.971 / 0.926	0.964 / 0.920

Table 2: Coverage probabilities of the upper confidence limit. For each combination of k and n, the nominal coverage probabilities ($= 1 - \alpha$) are 95% (left) and 90% (right).

Lastly, I considered the coverage probabilities of the confidence interval. The procedure is again the same, but this time I counted how many times the actual value of k lay in the confidence interval, i.e. between the lower confidence limit and the upper confidence limit. Also this time, I divided the number by 10.000 to get the coverage probabilities. In Table 3 the results can be seen.

n	**k = 0.05**	**k = 0.1**	**k = 0.5**	**k = 1**	**k = 1.5**
3	0.974 / 0.941	0.974 / 0.940	0.970 / 0.930	0.968 / 0.921	0.959 / 0.922
5	0.972 / 0.936	0.974 / 0.943	0.969 / 0.939	0.966 / 0.929	0.966 / 0.925
10	0.968 / 0.928	0.974 / 0.939	0.976 / 0.939	0.971 / 0.933	0.969 / 0.927

Table 3: Coverage probabilities of the confidence interval. For each combination of k and n, the nominal coverage probabilities ($= 1 - \alpha$) are 95% (left) and 90% (right).

As can be seen in each case, all the coverage probabilities are close to the nominal values (95%, and 90% respectively). Hence I can conclude that the method performs well, and I can conclude that the function in my R package is implemented correctly.

5.1.5 Comparison with the bootstrap method

In this section I compare the performance of the confidence intervals, created with the method described in section 5.1.2 (which I will now call the fiducial method), with the performance of the confidence intervals constructed using the bootstrap method. The bootstrap method is a widely used technique used to estimate statistics by averaging estimates from multiple data samples. One of these statistics can be the confidence interval. The exact method to compute confidence intervals using the bootstrap method is given below. [11]

Suppose a data set that is gam(k, θ) distributed is available. This data set is used to get an estimate of k and θ, using the method explained in Section 4.2. These two estimates, I call $\hat{k}$ and $\hat{\theta}$. Next, for a total of B (often $B = 2000$ or 10000) times, a realization of a a gam$(\hat{k}, \hat{\theta})$ distribution is generated with the same number of observations as the original gam(k, θ) distribution. These realizations are called bootstrap data sets. Of all these B bootstrap data sets, the bootstrap statistic is calculated. The bootstrap statistics I calculated were the parameters k and θ, which I call k^* and θ^*. This gives us a total of B estimates k^*'s and θ^*'s. Finally, to get an $(1 - \alpha)$ confidence interval, these estimates are sorted in ascending order and the $100\frac{\alpha}{2}$th and $100(1-\frac{\alpha}{2})$th percentiles are taken as the confidence interval for k or θ.

Using this method and using the same settings as defined in Section 5.1.4, coverage probabilities can be determined for the confidence interval, based on the bootstrap method. To compare the performance of the bootstrap method with the fiducial method, also the coverage probabilities for the confidence interval, based on the fiducial method need to be determined. For a good resemblance, I calculated the coverage probabilities for both the shape and the rate parameter.

In Table 4, the coverage probabilities of the confidence interval for the ***shape parameter*** are given, using the ***bootstrap method***. In Table 5, the same coverage probabilities are given for the ***shape parameter***, determined with ***fiducial method***. In Table 6, the coverage probabilities of the confidence interval for the ***rate parameter*** are given, using the ***bootstrap method***. Lastly, in Table 7, the same coverage probabilities are given for ***the rate parameter***, now determined with the ***fiducial method***. Note that for the bootstrap method, I used $B = 2000$, meaning I generated 2000 bootstrap data sets to calculate the confidence intervals.

n	k = 0.05	k = 0.1	k = 0.5	k = 1	k = 1.5
3	0.950 / 0.862	0.937 / 0.852	0.930 / 0.866	0.920 / 0.888	0.920 / 0.875
5	0.916 / 0.807	0.923 / 0.821	0.935 / 0.866	0.932 / 0.877	0.942 / 0.891
10	0.917 / 0.827	0.915 / 0.835	0.917 / 0.857	0.940 / 0.871	0.950 / 0.883

Table 4: Coverage probabilities of the confidence interval for the shape parameter, using the bootstrap method. For each combination of k and n, the nominal coverage probabilities ($= 1-\alpha$) are 95% (left) and 90% (right).

n	k = 0.05	k = 0.1	k = 0.5	k = 1	k = 1.5
3	0.973 / 0.939	0.972 / 0.940	0.969 / 0.930	0.964 / 0.925	0.960 / 0.923
5	0.971 / 0.930	0.971 / 0.943	0.969 / 0.939	0.972 / 0.928	0.968 / 0.921
10	0.967 / 0.927	0.974 / 0.940	0.973 / 0.937	0.969 / 0.937	0.965 / 0.925

Table 5: Coverage probabilities of the confidence interval for the shape parameter, using the fiducial method. For each combination of k and n, the nominal coverage probabilities ($= 1-\alpha$) are 95% (left) and 90% (right).

n	k = 0.05	k = 0.1	k = 0.5	k = 1	k = 1.5
3	0.471 / 0.427	0.648 / 0.601	0.875 / 0.855	0.893 / 0.880	0.901 / 0.884
5	0.575 / 0.546	0.755 / 0.690	0.912 / 0.891	0.933 / 0.905	0.947 / 0.934
10	0.710 / 0.658	0.817 / 0.778	0.936 / 0.926	0.952 / 0.922	0.956 / 0.910

Table 6: Coverage probabilities of the confidence interval for the rate parameter, using the bootstrap method. For each combination of k and n, the nominal coverage probabilities ($= 1-\alpha$) are 95% (left) and 90% (right).

n	k = 0.05	k = 0.1	k = 0.5	k = 1	k = 1.5
3	0.969 / 0.926	0.970 / 0.934	0.967 / 0.927	0.965 / 0.920	0.964 / 0.922
5	0.963 / 0.914	0.965 / 0.920	0.968 / 0.931	0.965 / 0.921	0.961 / 0.916
10	0.955 / 0.906	0.961 / 0.917	0.964 / 0.923	0.963 / 0.928	0.966 / 0.919

Table 7: Coverage probabilities of the confidence interval for the rate parameter, using the fiducial method. For each combination of k and n, the nominal coverage probabilities ($= 1-\alpha$) are 95% (left) and 90% (right).

Comparing Table 4 and 5, it can be seen that the fiducial method performs overall better than the bootstrap method, when constructing confidence intervals for the shape parameter. That is because the coverage probabilities in Table 5 are slightly higher. The same can be said for the confidence intervals for the rate parameter. Namely, comparing Table 6 and 7, it can be seen that the coverage probabilities of the bootstrap method are lower than the coverage probabilities determined with the fiducial method. This is especially noticeable when n and k are small.

In conclusion, the fiducial method proposed in Section 5.1.2 performs better than the bootstrap method, especially when the sample size or the shape parameter of a data set is small. Because of this, I will use the fiducial method in my R package, and not the often used bootstrap method.

5.2 Prediction limits

5.2.1 Introduction

Before explaining the relevancy of prediction limits, I will first state the definition of a prediction interval, a lower prediction limit and an upper prediction limit.

Definition: Assume n historical data, i.e. $x_1, ..., x_n$ from a gamma distribution are available. Consider two statistics $L = L(x_1, ..., x_n)$ and $U = U(x_1, ..., x_n)$ and a future measurement x_{n+1} from the same gamma distribution. If L, U and x_{n+1} have the following relationship

$$P(L \leq x_{n+1} <\leq U) = 1 - \alpha,$$

where $0 < \alpha < 1$ is a constant, then $[L, U]$ is an $(1 - \alpha)$ prediction interval for the single future measurement. When U equals ∞, L is called an $(1 - \alpha)$ lower prediction limit for the single future measurement. Lastly, when L equals $-\infty$, U is called an $(1 - \alpha)$ upper prediction limit for the single future measurement. [3]

In other words, the lower prediction limit provides an $(1-\alpha)$ confidence of a future measurement not being below a known number. On the other hand, the upper prediction limit provides an $(1 - \alpha)$ confidence of not being exceeded by the future measurements with a known number [3]. This is a very useful principle in for example the detection of contamination in ground water. Using historical data, the upper prediction limit can be compared with the next future measurement to check whether there is possible contamination. Upper prediction limits, but also lower prediction limits and prediction intervals, can in practice also be used in other areas, such as molecular genetics [3].

Prediction limits are very important to construct. To come back to the groundwater contamination example, a lot of contamination in groundwater may damage the environment and can be very dangerous for the health of humans and animals. This is one of the reasons why appropriate methods need to be found and implemented to construct these limits. In practice, there are already two methods to construct prediction limits for a gamma distribution. However both methods are found to be not appropriate when either the sample size is small or when the shape parameter k is small. That is why Piao Chen and Zhi-Sheng Ye have proposed a new method to construct prediction limits and intervals [3].

5.2.2 Main idea of the method

The first step of constructing prediction limits and intervals is to use Algorithm I again, explained in Appendix A. After B realizations of the shape parameter k and the rate parameter θ have been constructed, the following is done. For each $(k_b, \theta_b), b = 1, ..., B$, a random realization from a gam(k_b, θ_b) distribution is generated. This realization is denoted by PL_b. In this way, B values, $PL_1, ...PL_B$ are constructed.

To now construct an $(1 - \alpha)$ lower prediction limit, the values $PL_1, ..., PL_B$ are sorted in ascending order and the 100αth percentile is used as the $(1-\alpha)$ lower prediction limit for a single future measurement. Similarly, the $100(1 - \alpha)$th percentile of the sorted values $PL_1, ..., PL_B$ can be used as an $(1 - \alpha)$ upper prediction limit for a single future measurement.

Lastly, an $(1-\alpha)$ prediction interval can be obtained by taking the interval between the $(1 - \frac{\alpha}{2})$ lower confidence limit and the $(1 - \frac{\alpha}{2})$ upper confidence limit.

5.2.3 Implementation

In the first part of the function I created in my R package, I set up B realizations of the shape parameter k and the rate parameter θ. That part of the code is explained in Appendix A. After that, prediction limits and intervals are constructed in the way I explained in Section 5.2.2.

The function I created has as input the parameter *lim*. I made this, so that an user can ask for only a lower prediction limit, only an upper prediction limit or a prediction interval. The default value will always be a prediction interval. Lastly, when an user does not input a value for α and/or B, my function will also use a default value. For B that is 10.000, and for α, that is 0.05, sucht that a 95% prediction limit/interval is returned.

5.2.4 Validation

Again, I can check the performance of the method and my R function with coverage probabilities. I performed multiple Monte Carlo simulations, where I varied k in the range 0.05, 0.1, 0.5, 1, 1.5 and set $\theta = 1$. Once again, I consider small sample sizes, i.e. $n = 3, 5, 10$.

To check the performance of the lower prediction limit, I performed a Monte Carlo simulation, where in 10.000 replications, I counted how many times the lower prediction limit for a future measurement was indeed lower than the value of a future measurement. This number I divided by 10.000. In this way, I got the coverage probabilities of the lower prediction limit. In Table 8 my results are stated.

n	**k = 0.05**	**k = 0.1**	**k = 0.5**	**k = 1**	**k = 1.5**
3	0.969 / 0.917	0.969 / 0.932	0.967 / 0.924	0.961 / 0.920	0.960 / 0.921
5	0.960 / 0.913	0.962 / 0.920	0.961 / 0.921	0.959 / 0.916	0.959 / 0.911
10	0.957 / 0.908	0.957 / 0.906	0.958 / 0.914	0.956 / 0.905	0.953 / 0.907

Table 8: Coverage probabilities of the lower prediction limit. For each combination of k and n, the nominal coverage probabilities ($= 1 - \alpha$) are 95% (left) and 90% (right).

To check for the upper prediction limit I proceeded in the same way. Only this time, I counted how many times the upper prediction limit for a single future measurement was indeed higher than the value of a future measurement. Again, I divided that number by 10.000, to get the coverage probabilities. See Table 9 for my results.

n	**k = 0.05**	**k = 0.1**	**k = 0.5**	**k = 1**	**k = 1.5**
3	0.944 / 0.903	0.954 / 0.906	0.955 / 0.903	0.955 / 0.897	0.951 / 0.894
5	0.951 / 0.901	0.953 / 0.896	0.947 / 0.895	0.950 / 0.895	0.947 / 0.898
10	0.950 / 0.897	0.950 / 0.897	0.950 / 0.895	0.945 / 0.897	0.947 / 0.904

Table 9: Coverage probabilities of the upper prediction limit. For each combination of k and n, the nominal coverage probabilities ($= 1 - \alpha$) are 95% (left) and 90% (right).

Finally, to determine the coverage probabilities of the prediction interval, I counted how many times the actual value of the future measurement lay in the prediction interval, i.e. between the lower prediction limit and the upper prediction limit. Also now, I divided the number by 10.000 to get the coverage probabilities. These coverage probabilities can be seen in Table 10.

n	**k = 0.05**	**k = 0.1**	**k = 0.5**	**k = 1**	**k = 1.5**
3	0.972 / 0.923	0.972 / 0.929	0.963 / 0.916	0.959 / 0.916	0.962 / 0.910
5	0.962 / 0.917	0.964 / 0.919	0.962 / 0.917	0.957 / 0.911	0.956 / 0.902
10	0.947 / 0.906	0.956 / 0.901	0.955 / 0.901	0.954 / 0.906	0.949 / 0.898

Table 10: Coverage probabilities of the prediction interval. For each combination of k and n, the nominal coverage probabilities ($= 1 - \alpha$) are 95% (left) and 90% (right).

As can be seen in each case, all the coverage probabilities are close to the nominal values (95% and 90% respectively). Hence I can conclude that the method performs well, and I can conclude that the function in my R package is implemented correctly.

5.3 Tolerance limits

5.3.1 Introduction

When considering prediction limits, but the number of future measurements is either large or unknown, it is better to use a tolerance limit instead of a prediction limit. Before I explain more about tolerance limits/intervals, I will state the definitions of a lower tolerance limit, an upper tolerance limit and a tolerance interval.

Definition: Assume that n historical data, i.e. $x_1, ..., x_n$ from a gamma distribution with cumulative distribution function (CDF) F_{gam} are available. Consider two statistics $L = L(x_1, ..., x_n)$ and $U = U(x_1, ..., x_n)$. Let β and α be two constants and $0 < \beta, \alpha < 1$. If L and U are determined such that

$$P(F_{gam}(U) - F_{gam}(L) \geq \beta) = 1 - \alpha,$$

then $[L, U]$ is called a $(\beta, 1-\alpha)$ two-sided tolerance interval. When U equals ∞, L is called a $(\beta, 1-\alpha)$ lower tolerance limit if

$$P(1 - F_{gam}(L) \geq \beta) = 1 - \alpha.$$

Similarly, when L equals $-\infty$, U is called a $(\beta, 1-\alpha)$ upper tolerance limit if

$$P(F_{gam}(U) \geq \beta) = 1 - \alpha \text{ [12]}.$$

In other words, a $(\beta, 1-\alpha)$ tolerance interval should contain a proportion of at least β of the future measurements with $(1-\alpha)$ confidence. In this way, a $(\beta, 1-\alpha)$ lower tolerance limit provides a $(1-\alpha)$ confidence of a proportion β of future measurements not being below a known number. Similarly, a $(\beta, 1-\alpha)$ upper tolerance limit provides a $(1-\alpha)$ confidence of a proportion β of future measurements not being above a known number. In this way, just like the upper prediction limit, the upper tolerance limit is very useful for the detection of possible contamination of ground water. Only this time, the upper tolerance limit is compared with a proportion of future measurements to detect possible contamination. Besides the detection of contamination in ground water, tolerance limits and tolerance intervals are also used in other areas, such as for the purpose of controlling product quality [3].

Because of the same environmental and health reasons as I gave for prediction limits, tolerance limits are very important to construct. However, the two existing methods do not perform well when either the sample size is small or when the shape parameter k is small. That is why Piao Chen and Zhi-Sheng Ye proposed a new method to construct tolerance limits and intervals [3].

5.3.2 Main idea of the method

The method makes use of the following property of tolerance limits. Namely, a $(\beta, 1-\alpha)$ upper tolerance limit is equal to an $(1-\alpha)$ upper confidence limit for the βth quantile τ_β. Also, a $(\beta, 1-\alpha)$ lower tolerance limit is equal to an $(1-\alpha)$ lower confidence limit for the $(1-\beta)$th quantile $\tau_{1-\beta}$. With this knowledge, the algorithm is given as follows.

Just like with the constructing confidence- and prediction limits, also the algorithm of constructing tolerance limits starts by using Algorithm I, explained in Appendix A, to first construct B realizations of the shape parameter k and rate parameter θ.

For constructing a lower tolerance limit, B $(1-\beta)$th quantiles $\tau_{1-\beta}$ of a gamma distribution with parameters k_b and θ_b are computed, denoted as $\tau_{1-\beta_b}$. Doing this gives a list of B values $\tau_{1-\beta_1}, ..., \tau_{1-\beta_B}$. Sorting these values in ascending order and using the 100αth percentile, gives the $(\beta, 1-\alpha)$ lower tolerance limit.

Constructing the upper tolerance limit goes in a similar way. First B βth quantiles τ_β of a gamma distribution with parameters k_b and θ_b are computed, denoted as τ_{β_b}. Then a list is made of these B values $\tau_{\beta_1}, ..., \tau_{\beta_B}$. And finally, sorting these values in ascending order and using the $100(1-\alpha)$th percentile, gives the $(\beta, 1-\alpha)$ upper tolerance limit.

To construct the tolerance interval, I make use of the WH approximation, based on the normal distribution. This method is proposed by K. Krishnamoorthy, Thomas Mathew and Shubhabrata Mukherjee [4]. The method starts by assuming that a random variable X with observations $x_1, ..., x_n$ is gam(k, θ) distributed. Wilson and Hilferty (1931) [13] argued that $X^{1/3} \sim \mathcal{N}(\mu, \sigma^2)$ approximately, with

$$\mu = \frac{\Gamma(k+1/3)}{\Gamma(k)} \quad \text{and} \quad \sigma^2 = \frac{\Gamma(k+2/3)}{\Gamma(k)} - \mu^2,$$

where k is the shape parameter of the gamma distribution and $\Gamma(\cdot)$ is the gamma function, defined in Section 2.

In this way, let $X_1, ..., X_n$ be observations from a gam(k, θ) distribution. Let $Y = X^{1/3}$ be normally distributed as explained above. Define

$$\bar{Y} = \frac{1}{n}\sum_{i=1}^{n} Y_i \quad \text{and} \quad S_y^2 = \frac{1}{n-1}\sum_{i=1}^{n}(Y_i - \bar{Y})^2.$$

Then a $(\beta, 1-\alpha)$ two-sided tolerance interval for the normally distributed random variable Y is given by

$$\bar{Y} \pm cS_y, \quad \text{with} \quad c = \left(\frac{(n-1)\chi^2_{1,\beta}(1/n)}{\chi^2_{n-1,\alpha}}\right)^{1/2} \tag{9}$$

In this formula, $\chi^2_{1,\beta}(1/n)$ denotes the βth quantile of a non-central chi-squared distribution with degrees of freedom 1 and non-centrality parameter $1/n$. The αth quantile of a central chi-squared distribution with $n-1$ degrees of freedom is denoted by $\chi^2_{n-1,\alpha}$.

As said before, if X is gam(k, θ) distributed, then $X^{1/3} = Y$ is normally distribution. Inverting this relation, yields that Y^3 is again gamma distributed. Using this, the $(\beta, 1-\alpha)$ two-sided tolerance interval for the gamma distribution is given by

$$(\bar{Y} \pm cS_y)^3,$$

with c defined in (9). Note that since a normal distribution can take negative values, the expression $\bar{Y} \pm cS_y$ of a lower tolerance limit for a normally distributed random variable can also be negative. In this way, it is possible that $(\bar{Y} \pm cS_y)^3$ is negative, meaning that the lower tolerance limit for the gamma distribution can take a negative value. However, the gamma distribution is only defined for positive values. To fix this problem, the lower tolerance limit for a gamma distribution is set to be 0, when $(\bar{Y} \pm cS_y)^3$ turns out to be negative.

5.3.3 Implementation

The function I created has as input the parameter *lim*. I made this, so that an user can ask for only a lower tolerance limit, only an upper tolerance limit or a tolerance interval. When the user asks for either a lower or upper tolerance limit, I first set up B realizations of k and θ. That part of the code is explained in Appendix A. After that, lower and upper tolerance limits are constructed in the way explained in Section 5.3.2. When the user asks for a tolerance interval, the WH approximation, described in Section 5.3.2, is used to construct the tolerance interval. By default a tolerance interval will be returned

Lastly, when an user does not input a value for α, β and/or B, my function will also use a default value. For B that is 10000, for α that is 0.05 and for β that is 0.99, such that a (99%, 95%) tolerance interval is returned.

5.3.4 Validation

Also the performance of the tolerance limits and of my R function, I can check using coverage probabilities. Again I performed multiple Monte Carlo simulations. The shape parameter k was again equal to 0.05, 0.1, 0.5, 1, 1.5, while θ was set at 1. I considered small sizes, this time $n = 5, 10, 15$. Furthermore, I let $\alpha = 0.01, 0.05, 0.10$ such that $1 - \alpha = 0.99, 0.95, 0.90$. Lastly, $\beta = 0.90, 0.95, 0.99$.

To check the performance of the lower tolerance limit, I performed a Monte Carlo simulation, where in 2000 replications, I counted how many times the lower tolerance limit was indeed lower than the value of the $(1 - \beta)$th quantile. This number was divided by 2000, to get the coverage probabilities. In Table 11 I stated my results.

		$\beta = 0.90$			$\beta = 0.95$			$\beta = 0.99$		
n	**k**	**0.99**	**0.95**	**0.90**	**0.99**	**0.95**	**0.90**	**0.99**	**0.95**	**0.90**
5	**0.05**	0.999	0.976	0.944	0.999	0.981	0.939	0.997	0.982	0.922
	0.1	0.999	0.988	0.942	0.999	0.979	0.939	0.999	0.982	0.950
	0.5	0.996	0.974	0.938	0.995	0.971	0.925	0.993	0.971	0.937
	1	0.992	0.964	0.914	0.992	0.963	0.910	0.996	0.967	0.925
	1.5	0.992	0.961	0.926	0.992	0.963	0.922	0.994	0.956	0.912
10	**0.05**	0.997	0.971	0.924	0.996	0.971	0.934	0.998	0.978	0.936
	0.1	0.998	0.976	0.928	0.998	0.980	0.940	0.999	0.974	0.939
	0.5	0.996	0.976	0.927	0.998	0.975	0.936	0.996	0.979	0.944
	1	0.994	0.966	0.919	0.994	0.964	0.921	0.994	0.953	0.926
	1.5	0.994	0.958	0.921	0.994	0.961	0.915	0.993	0.962	0.922
15	**0.05**	0.997	0.971	0.924	0.998	0.968	0.925	0.997	0.966	0.933
	0.1	0.996	0.979	0.947	0.997	0.972	0.922	0.999	0.969	0.936
	0.5	0.996	0.970	0.940	0.996	0.971	0.934	0.997	0.975	0.924
	1	0.990	0.971	0.916	0.994	0.966	0.914	0.997	0.960	0.924
	1.5	0.993	0.969	0.921	0.996	0.961	0.928	0.993	0.966	0.916

Table 11: Coverage probabilities of the lower tolerance limit for each combination of n, k, α and β.

To check for the upper tolerance limit I did the same thing. Only now, I counted how many times the upper tolerance limit was indeed higher than the value of the βth quantile, and divided this number by 2000. This gave me again the coverage probabilities. In Table 12, my results can be seen.

		$\beta = 0.90$			$\beta = 0.95$			$\beta = 0.99$		
n	**k**	**0.99**	**0.95**	**0.90**	**0.99**	**0.95**	**0.90**	**0.99**	**0.95**	**0.90**
5	**0.05**	0.995	0.956	0.911	0.994	0.963	0.926	0.996	0.947	0.899
	0.1	0.995	0.950	0.909	0.998	0.963	0.909	0.997	0.959	0.913
	0.5	0.993	0.956	0.896	0.992	0.958	0.912	0.993	0.959	0.904
	1	0.991	0.949	0.896	0.993	0.952	0.907	0.995	0.962	0.900
	1.5	0.989	0.948	0.892	0.993	0.945	0.908	0.992	0.952	0.907
10	**0.05**	0.993	0.946	0.904	0.992	0.951	0.900	0.992	0.949	0.900
	0.1	0.990	0.954	0.905	0.991	0.952	0.901	0.994	0.955	0.897
	0.5	0.987	0.948	0.892	0.994	0.961	0.900	0.991	0.961	0.910
	1	0.988	0.948	0.892	0.991	0.955	0.902	0.994	0.956	0.909
	1.5	0.989	0.948	0.880	0.989	0.950	0.903	0.989	0.957	0.898
15	**0.05**	0.988	0.959	0.897	0.992	0.952	0.895	0.994	0.952	0.893
	0.1	0.986	0.957	0.898	0.996	0.949	0.912	0.992	0.956	0.902
	0.5	0.988	0.956	0.898	0.996	0.949	0.912	0.992	0.956	0.902
	1	0.987	0.955	0.911	0.993	0.956	0.886	0.993	0.959	0.898
	1.5	0.987	0.953	0.891	0.988	0.952	0.910	0.993	0.956	0.905

Table 12: Coverage probabilities of the upper tolerance limit for each combination of n, k, α and β.

To construct the coverage probabilities of the tolerance interval, I make use of another definition of the two-sided tolerance interval. Note that this is just a special case of the definition I gave before.

Definition: Assume that n historical data, i.e. $x_1, ..., x_n$ from a gamma distribution are available. Consider two statistics $L = L(x_1, ..., x_n)$ and $U = U(x_1, ..., x_n)$. Let β and α be two constants and $0 < \beta, \alpha < 1$. Then $[L, U]$ is called a $(\beta, 1-\alpha)$ two sided tolerance interval, if

$$P(L \leq \tau_{(1-\beta)/2} \text{ and } \tau_{(1+\beta)/2} \leq U) = 1 - \alpha, \tag{10}$$

where τ_β is the βth quantile of a gamma distribution.

To now construct coverage probabilities, I considered the same α's and β's as before. But k was taken as 0.5, 1.5, 3, 9 and n was taken as 3, 7, 12. This is because, the WH approximation does not work well for a small shape parameter k. To compute the coverage probabilities I counted how many times both the lower tolerance limit was smaller than a $(1-\beta)/2$ quantile and the upper tolerance limit was bigger than a $(1+\beta)/2$ quantile. This number was divided by 2000. The coverage probabilities are stated in Table 13. Note that the performance gets better when n becomes smaller.

		$\beta = 0.90$			$\beta = 0.95$			$\beta = 0.99$		
n	**k**	**0.99**	**0.95**	**0.90**	**0.99**	**0.95**	**0.90**	**0.99**	**0.95**	**0.90**
3	**0.5**	0.985	0.937	0.873	0.988	0.943	0.886	0.987	0.945	0.901
	1.5	0.989	0.940	0.876	0.989	0.946	0.885	0.990	0.950	0.895
	3	0.987	0.935	0.874	0.989	0.945	0.887	0.989	0.949	0.893
	9	0.988	0.939	0.875	0.989	0.939	0.886	0.989	0.945	0.891
7	**0.5**	0.979	0.909	0.833	0.982	0.925	0.852	0.988	0.942	0.879
	1.5	0.983	0.915	0.844	0.983	0.925	0.866	0.988	0.948	0.892
	3	0.980	0.914	0.834	0.982	0.926	0.857	0.988	0.937	0.881
	9	0.982	0.913	0.830	0.983	0.928	0.856	0.988	0.935	0.876
12	**0.5**	0.966	0.883	0.809	0.977	0.899	0.828	0.980	0.922	0.859
	1.5	0.977	0.898	0.826	0.982	0.918	0.846	0.989	0.941	0.881
	3	0.974	0.892	0.816	0.981	0.916	0.841	0.987	0.938	0.870
	9	0.975	0.902	0.803	0.981	0.907	0.834	0.985	0.929	0.864

Table 13: Coverage probabilities of the tolerance interval for each combination of n, k, α and β.

As can be seen in each case, all the coverage probabilities are close to the nominal values (99%, 95% and 90% respectively). Hence I can conclude that the method performs well, and I can conclude that the function in my R package is implemented correctly.

6 Illustrative example

In this section, I use real data to illustrate the application of my R package. The data set that I use is a data set of observations on the amount of precipitation (rainfall) in ml/m^2 during 227 storms in Illinois from 1960 and 1964 [14]. This data set is given in Table 14.

0.020	0.001	0.001	0.120	0.080	0.420	1.720	0.050	0.010	0.010	0.003	0.001
0.003	0.270	0.001	0.060	0.050	2.130	0.040	1.100	0.020	0.001	0.140	0.080
0.210	0.070	0.320	0.240	0.290	0.001	0.290	1.130	0.003	0.010	0.190	0.002
0.010	0.040	0.002	0.070	0.450	0.010	0.180	0.670	0.003	0.010	0.040	0.002
0.490	0.020	0.020	0.340	0.140	0.370	0.330	0.330	0.350	0.010	0.500	0.760
1.060	0.002	0.060	0.160	0.270	0.250	0.290	0.020	0.050	0.460	0.070	0.410
0.020	0.080	0.210	0.010	0.440	0.020	0.050	0.110	1.500	0.003	0.180	0.010
0.002	0.240	0.010	0.750	0.010	0.140	0.130	0.010	0.010	0.270	0.450	1.780
0.250	0.240	0.004	0.210	0.170	0.830	0.150	0.030	0.030	0.500	0.040	0.090
0.040	0.060	0.060	0.120	0.003	0.003	0.400	0.020	0.510	0.003	0.020	0.020
0.020	0.010	0.001	0.140	0.001	0.100	0.010	1.090	0.010	0.002	0.001	0.840
0.030	0.350	0.070	0.001	0.002	0.002	0.200	0.060	0.140	0.010	0.020	0.020
0.002	0.001	0.550	0.130	0.190	2.100	0.090	0.350	0.790	0.320	1.350	0.170
0.020	0.002	0.010	0.250	0.230	0.170	0.010	0.020	0.001	0.010	0.020	0.110
0.210	1.260	0.010	0.730	0.100	0.090	0.007	0.360	0.770	0.210	1.270	0.070
0.080	0.160	0.260	0.010	0.230	0.080	0.020	0.010	0.290	0.010	0.010	0.070
0.400	0.002	0.003	0.010	0.090	0.160	0.040	0.270	0.730	0.410	0.030	0.120
0.030	1.040	0.060	0.090	0.730	0.040	0.160	0.590	0.003	0.002	0.020	0.004
0.010	0.001	0.060	0.620	0.010	0.520	0.110	0.003	0.600	0.002	0.050	

Table 14: Rainfall data of 227 storms in Illinois (1960-1964).

First of all, I use the function *MLest(x)* to estimate the model parameters (shape parameter k, rate parameter θ and scale parameter β) of the data set. These are given by $k \approx 0.415, \theta \approx 1.815$ and $\beta \approx 0.536$.

```
x = Rainfall
MLest(x)
$shape
[1] 0.4153486

$rate
[1] 1.850995

$scale
[1] 0.5355317
```

Next, I use the function *conflimits(x)* to determine the 95% confidence intervals for k, θ and β. For the shape parameter k, the 95% confidence interval is given by approximately (0.370, 0.522), while the estimated shape parameter was given by approximately 0.415. Furthermore, for the rate parameter θ, the 95% confidence interval is given by approximately (1.476, 2.511), while the estimated rate parameter was given by approximately 1.851. Lastly, for the scale parameter β, the 95% confidence interval is given by approximately (0.396, 0.668), while the estimated shape parameter was given by approximately 0.536.

```
conflimits(x, par = "shape")
$lower confidence limit for the shape parameter
     2.5%
0.3700199

$upper confidence limit for the shape parameter
    97.5%
0.5223945

conflimits(x, par = "rate")
$lower confidence limit for the rate parameter
    2.5%
1.475686

$upper confidence limit for the rate parameter
   97.5%
2.510914

conflimits(x, par = "scale")
$lower confidence limit for the scale parameter
     2.5%
0.3962517

$upper confidence limit for the scale parameter
    97.5%
0.6680313
```

After the confidence intervals have been determined, the prediction interval will be estimated. I do this by using the function *predlimits(x)*. The 95% is given by approximately $(0.0000975, 1.215)$. That means that a future measurement will be above 0.0000975 and below 1.215, with 95% confidence.

```
predlimits(x)
$lower prediction limit
        2.5%
9.750822e-05

$upper prediction limit
   97.5%
1.214974
```

Lastly, the tolerance interval will be calculated, by using the *tollimits(x)* function. The (99%, 95%) tolerance interval is given by approximately (0, 1.973). That means that 99% of future measurement will be bigger than zero and lower than 1.973, with 95% confidence.

```
tollimits(x)
$lower tolerance limit
[1] 0

$upper tolerance limit
[1] 1.972505
```

7 Conclusion

During this project, I managed to create an R package for the gamma distribution. In this package I created five functions. The function *rGamma* generates gamma distributed random variables. This performance of this function is tested by calculating the acceptance rate for different values of k between 0.001 and 1. An average acceptance rate of around 0.710 was found, which indicates that the method and the implementation is correct. The second function I created was *MLest* to estimate the model parameters of a gamma distribution. The performance of the newly proposed methods is roughly the same as the maximum likelihood estimators. Also, the performance of the proposed estimators was tested against the method of moments estimators. In terms of rMSE, the newly proposed estimators performed better. From this, I concluded that I preferred the proposed estimators over the maximum likelihood estimators and the method of moments estimators.

The last three functions were constructed to determine the confidence, prediction and tolerance limits/intervals of a gamma distribution. These methods were tested using coverage probabilities. Since all these coverage probabilities were close to the nominal values, the performance of the constructed methods were good, meaning that the implementation of the methods is correct. Also, the used method (fiducial method) to construct confidence intervals was tested against bootstrap method. Especially, when the sample size and shape parameter were small, the performance of the fiducial method was much better than the bootstrap method.

All in all, I created an R package that works for data that is gamma distributed. As said above, all the functions have been tested on their performance. Lastly, the functions have been tested with a real data set, to show that the package works in practice.

8 Discussion

All the functions and methods I created, but one, were build in the way I initially had in mind. Only the method to construct tolerance intervals is implemented differently than was intended. The method I originally wanted to use, is the method described by Piao Chen and Zhi-Sheng Ye [3]. However, after trying to implement it for weeks, I did not obtain the results that I wanted. As alternative, I used the WH-approximation to construct tolerance intervals, proposed by K. Krishnamoorthy, T. Mathew, and S. Mukherjee [4]. This method was easier to implement. However, when calculating the coverage probabilities of the tolerance interval, using this method, I noticed something strange. The performance of the method got worse when the number of observations was increased. Also, this the WH-approximate does not work well for a gamma distribution if $k < 0.5$. That is why the tolerance limits function is one of aspects that could be investigated in further research.

Another part that can also be improved, is the Monte Carlo simulations I have done for the tolerance limits and intervals. Because the algorithms to construct tolerance limits and intervals were a bit complex, the running time was quite long. That is why, I choose to do 2000 Monte Carlo simulations to determine the coverage probabilities. However, a choice of 10.000 would be more representative and could be done in future research to give more accurate results.

In addition to these programming improvements, there is also room for more tests and validations. In order to construct confidence intervals I have tested the performance of the fiducial method against the bootstrap method. I can also do these kind of tests for other methods. For example, I can try to implement a different method to generate gamma distributed random variables, and see which method works the best.

What can also be done in future research, is to expand the package by constructing more functions that are applicable for the gamma distribution. For example, the current package is able to construct gamma distributed random variables, but it would also be interesting to look into the constructing of gamma quantiles.

Also, the package can be enlarged by constructing the same functions for the generalized gamma distribution, since many often-used distributions (Gamma, Exponential, Weibull) are special cases of the generalized gamma distribution. This makes the generalized gamma distribution an interesting model to investigate.

References

[1] Xi, B., Tan, K. and Liu, C., 2013. "Logarithmic Transformation-Based Gamma Random Number Generators." Journal of Statistical Software, Volume 55, Issue 4, pp.1-17. Available at: https://www.jstatsoft.org/article/view/v055i04.

[2] Ye, Z. and Chen, N., 2017. "Closed-Form Estimators For The Gamma Distribution Derived From Likelihood Equations." The American Statistician, Volume 71, Number 2, pp.177-181. Available at: https://amstat.tandfonline.com/doi/abs/10.1080/00031305.2016.1209129

[3] Chen, P. and Ye, Z., 2017. "Approximate Statistical Limits For A Gamma Distribution." Journal of Quality Technology, 49, 1; ABI/INFORM Collection, pp.64-77. Available at: https://www.tandfonline.com/doi/abs/10.1080/00224065.2017.11918185.

[4] Krishnamoorthy, K. Mathew, T. Mukherjee, S. 2008. "Normal-Based Methods for a Gamma Distribution: Prediction and Tolerance Intervals and Stress-Strength Reliability". Technometrics 50(1), pp. 69-78. Available at: https://faculty.washington.edu/smukherj/mypubs/Gamma_normal_methods_2008.pdf.

[5] ROC Science. (n.d). "Gamma Distribution In A Roctopple Probabilistic Analysis." Available at: https://www.rocscience.com/help/roctopple/roctopple/Gamma_Distribution.htm.

[6] Cook, J.D., 2014. "Diagram of distribution relationships". [Blog] Available at: https://www.johndcook.com/blog/distribution_chart/.

[7] H. Pishro-Nik. 2014. "Introduction to probability, statistics, and random processes". Section 4.2.4 Gamma distribution. Kappa Research LLC. Available at: https://www.probabilitycourse.com.

[8] Geyer, C.J. 2003. "Stat 5101 Notes: Brand Name Distributions page 9-10." [Notes] Available at: http://www.stat.umn.edu/geyer/old03/5102/notes/brand.pdf.

[9] Kim, A., 2019. "Gamma Distribution - Intuition, Derivation, and Examples and why does it matter?" [Blog] Available at: https://towardsdatascience.com/gamma-distribution-intuition-derivation-and-examples-55f407423840.

[10] Ahrens, JH, Dieter, U. 1974. "Computer Methods for Sampling from Gamma, Beta, Poisson and Binomial Distributions." Computing, 12(3), pp. 223–246. Available at: https://link.springer.com/article/10.1007%2FBF02293108.

[11] Chen, P., 2020. Probability And Statistics: Lecture 12. Parametric Bootstrap.

[12] NCSS Statistical Software, (n.d). Tolerance Intervals. pp.585-1. Available at: ttps://www.ncss.com/wp-content/themes/ncss/pdf/Procedures/NCSS/Tolerance_Intervals.pdf.

[13] Wilson, E. B., and Hilferty, M. M. 1931. "The Distribution of Chi-Squares." Proceedings of the National Academy of Sciences, 17, 684–688. Available at: https://www.pnas.org/content/17/12/684.

[14] Rice, J.A., 2007. Mathematical Statistics And Data Analysis. 3rd ed. Duxbury Advanced Series. Available at: https://www.academia.edu/39655119/Mathematical_Statistics_and_Data_Analysis.

[15] Bain, L. and Engelhardt, M. 1975. "A Two-Moment Chi-Squared Approximation for the statistic $\log(\bar{x}/\tilde{x})$". Journal of the American Statistical Association 70(352), pp. 948-950. Available at: https://www.tandfonline.com/doi/abs/10.1080/01621459.1975.10480328.

A Algorithm I: Obtain realizations of (k, θ)

In this section I will provide the algorithm to obtain B realizations of k and θ. This algorithm I use in the constructing of confidence limits/intervals, prediction limits/intervals and tolerance limits.

First recall the probability density function of a random variable X that is gam(k, θ) distributed. It is given by

$$f_{gam}(x) = \frac{\theta^k}{\Gamma(k)} x^{k-1} e^{-\theta x}, x > 0,$$

where k is the shape parameter and θ is the rate parameter.

Let $x_1, x_2, ..., x_n$ be a sample from gam(k, θ). In 1975, Bain and Engelhardt [15] assumed that

$$W(k) = 2nk\log(\bar{x}/\tilde{x}) \sim c\chi^2_{(v)},$$

where $\bar{x} = \sum_i x_i/n$ and $\tilde{x} = \prod_i x_i^{1/n}$. Furthermore $\chi^2_{(v)}$ is the χ^2 distribution with v degrees of freedom.

The values of c and v are still to be determined, and that can be done by the moment matching method. Since a χ^2 distribution with v degrees of freedom has expectation equal to v and variance equal to $2v$, it is known that

$$\mathbb{E}(W) = cv \text{ and } \text{Var}(W) = 2c^2v.$$

Next, define $S_0 = \bar{x}/\tilde{x}$ and $S_1 = \log(S_0)$. This yields that

$$\mathbb{E}(S_1) = -\log(n) + \psi(kn) - \psi(k) \text{ and } \text{Var}(S_1) = -\psi_1(kn) + \psi_1(k)/n,$$

where $\psi(x)$ is the digamma function given by $\psi(x) = \frac{d\log\Gamma(x)}{dx}$ and $\psi_1(x)$ is the trigamma function given by $\psi_1(x) = \frac{d^2\log\Gamma(x)}{dx^2}$. Since $W = 2nkS_1$, it holds that $\mathbb{E}(W) = 2nk\mathbb{E}(S_1)$ and $\text{Var}(W) = 4n^2k^2\text{Var}(S_1)$. Comparing this with the expectation and variance found before, gives the system

$$\begin{aligned} \mathbb{E}(W) &= 2nk\mathbb{E}(S_1) = cv, \\ \text{Var}(W) &= 4n^2k^2\text{Var}(S_1) = 2c^2v. \end{aligned}$$

Solving this system gives the following expressions for v and c

$$v(k) = 2\mathbb{E}^2(W)/\text{Var}(W), \tag{11}$$
$$c(k) = \mathbb{E}(W)/v. \tag{12}$$

Note that c and v are functions of the unknown shape parameter k. The maximum likelihood estimator of k can be substituted to get values of c and v. Unfortunately, the maximum likelihood estimator of k does not have a closed form. However, a new estimator of k, estimated in the same way as in Section 4, that does have a closed form is

$$\hat{k} = \frac{(n-1)\sum_{i=1}^n x_i}{n\sum_{i=1}^n x_i\log(x_i) - \sum_{i=1}^n \log(x_i)\sum_{i=1}^n x_i}. \tag{13}$$

After obtaining c and v using $\hat{k}$, the realizations of k and θ can be obtained. Start by defining $U_0 \sim c\chi^2_{(v)}$. Because $W(k)$ is a monotone function of k and $W(k)$ has the same distribution as U_0, the exact form for k can be obtained as

$$k = \frac{U_0}{2n\log(\bar{x}/\tilde{x})}. \tag{14}$$

To determine θ , let $Z(\theta) = 2\theta\sum_i x_i$ and $U_1 \sim \chi^2_{(2nk)}$. Since $Z(\theta)$ and U_1 have the same distribution [3] and because $Z(\theta)$ is a monotone function of θ, θ can be obtained as

$$\theta = \frac{U_1}{2\sum_i x_i} \tag{15}$$

Summarizing everything above gives the following algorithm to obtain B realizations of the shape parameter k and the rate parameter θ.

Start with a data set set $x_1, ..., x_n$ that is gamma distributed. Compute the maximum likelihood estimator of k by Equation (13), denoted as $\hat{k}$. This $\hat{k}$ can be substituted in Equations (11) and (12) to get the values of c and v, denoted as $\hat{c}$ and $\hat{v}$, respectively. Generate B different U_0's, by using a chi-squared distribution and substitute these U_0's into Equation (14) to get $k_1, ..., k_B$. Then generate B different U_1's using a chi-squared distribution and the already found $k_1, ..., k_B$. Substitute these U_1's into Equation (15) to get $\theta_1, ..., \theta_B$.