Assignment 7

EMATM0061: Statistical Computing and Empirical Methods, TB1, 2022

Introduction

This is the seventh assignment for Statistical Computing and Empirical Methods (Unit EMATM0061) on the MSc in Data Science & MSc in Financial Technology with Data Science. This assignment is mainly based on Lectures 16, 17 and 18 (see the Blackboard).

The submission deadline for this assignment is 23:59, 21 November 2022. Note that this assignment will not count towards your final grade. However, it is recommended that you try to answer the questions to gain a better understanding of the concepts.

Create an R Markdown for the assignment

It is a good practice to use R Markdown to organize your code and results. You can start with the template called `Assignment07_Template.Rmd` which can be downloaded via Blackboard.

If you are considering submitting your solutions, please generate a PDF file. For example, you can choose the "PDF" option when creating the R Markdown file (note that this option may require Tex to be installed on your computer), or use R Markdown to output an HTML and print it as a PDF file in a browser, or use your own way of creating a PDF file that contains your solutions.

Only a PDF file will be accepted in the submission of this assignment. To submit the assignment, please visit the "Assignment" tab on the Blackboard page, where you downloaded the assignment.

Wish to know more about a perticular question?

You may want to ask a question during the computer lab.

Alternatively, we are collecting questions about this assignment that need to be addressed, through the following form. And this can be done either during the labs or outside the lab sessions. So, If you found a question in this assignment interesting but had difficulty in a particular step when trying to develop your answer, please put your remark in the form via the following link. A brief description of the difficulty would be very helpful. Giving your remark is optional, but we aim to know the most common questions that you might want to get some support.

https://forms.office.com/r/qggZ6UgWVb

Load packages

Some of the questions in this assignment require the tidyverse package. If it hasn't been installed on your computer, please use `install.packages()` to install them first.

To road the tidyverse package:

```
library(tidyverse)
```

1. Maximum likelihood estimates

In this section, we will explore maximum likelihood estimates that was introduced in Lecture 16.

1.1 Maximum likelihood estimates for Red tailed hawks

In this question we will fit a Gaussian model to a Red-Tailed hawk data set. First load the Hawks data set as follows:

```r
library(Stat2Data)
data("Hawks")
```

(Q1) Now use your data wrangling skills to filter extract a subset of the Hawks data set so that every Hawk belongs to the "`Red-Tailed`" species, and extract the "Weight", "Tail" and "Wing" columns. The returned output should be a data frame called "RedTailedDf" with three numerical columns and 577 examples.

Answer

```r
RedTailedDf <- Hawks %>% filter(Species=="RT") %>%
  select(Weight, Tail, Wing)
```

Display the first five rows of the "RedTailedDf". The resulting subset of the data frame should look as follows:

```
##   Weight Tail Wing
## 1    920  219  385
## 2    930  221  376
## 3    990  235  381
## 4   1090  230  412
## 5    960  212  370
```

(Q2)

We now model the vector of tail lengths from "RedTailedDf" as a sequence $X_1, \cdots, X_n \sim \mathcal{N}(\mu_0, \sigma_0^2)$ consisting of independent and identically distributed with unknown population mean μ_0 and population variance σ_0^2.

The maximum likelihood estimates for μ_0 is given by $\hat{\mu}_{\text{MLE}} = \frac{1}{n}\sum_{i=1}^{n} X_i$ and the maximum likelihood estimate for σ_0^2 is given by $\hat{\sigma}^2_{\text{MLE}} = \frac{1}{n}\sum_{i=1}^{n}(X_i - \hat{\mu}_{\text{MLE}})^2$.

Apply the maximum likelihood method to compute the estimates $\hat{\mu}_{\text{MLE}}$ and $\hat{\sigma}^2_{\text{MLE}}$ for tail lengths using the sample in "RedTailedDf"

Answer:

```r
n = length(RedTailedDf$Tail)
mu_mle <- mean(RedTailedDf$Tail, na.rm=TRUE)
sigma_mle <- sd(RedTailedDf$Tail, na.rm=TRUE) * sqrt((n-1)/n)
sigma_squared_mle <- sigma_mle^2;
```

(Q3)

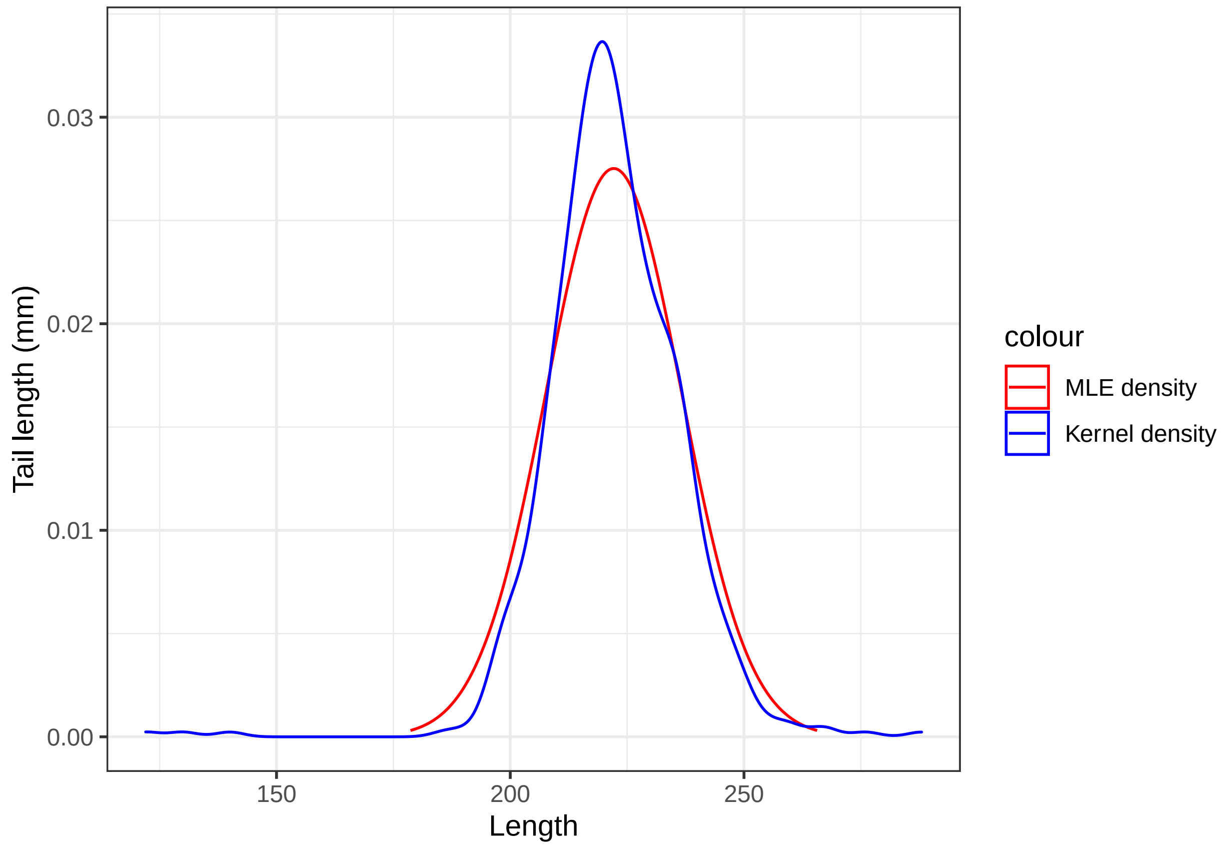
Next generate a plot which compares the probability density function for your fitted Gaussian model for the tail length of the Red-Tailed hawks with a kernel density plot.

Your plot should look as follows:

Answer

```r
# indices
len <- seq(mu_mle-3*sigma_mle, mu_mle+3*sigma_mle, sigma_mle*0.001)

# plot estimated density function
color <- c("MLE density"="red", "Kernel density"="blue")
estimated_density <- data.frame(Length=len, Density=dnorm(len, mean=mu_mle, sd=sigma_mle))
```

```r
plot_obj <- ggplot() + geom_line(data=estimated_density,
                                 aes(x=Length, y=Density, color="MLE density"))

# kernel density plot of the sample
plot_obj + geom_density(data=RedTailedDf,
                        aes(x=Tail, color="Kernel density")) +
  labs(y="Tail length (mm)") +
  theme_bw() + scale_color_manual(values = color)
```



1.2 Unbiased estimation of the population variance

In this question we consider i.i.d. samples $X_1, \cdots, X_n \sim \mathcal{N}(\mu_0, \sigma_0^2)$ with unknown population mean μ_0 and unknown population variance σ_0^2.

Let $\overline{X}$ be the sample mean, Let $\hat{V}_{\text{MLE}} = \frac{1}{n}\sum_{i=1}^{n}(X_i - \overline{X})^2$ and let $\hat{V}_U := \frac{1}{n-1}\sum_{i=1}^{n}(X_i - \overline{X})^2$

(Q1)

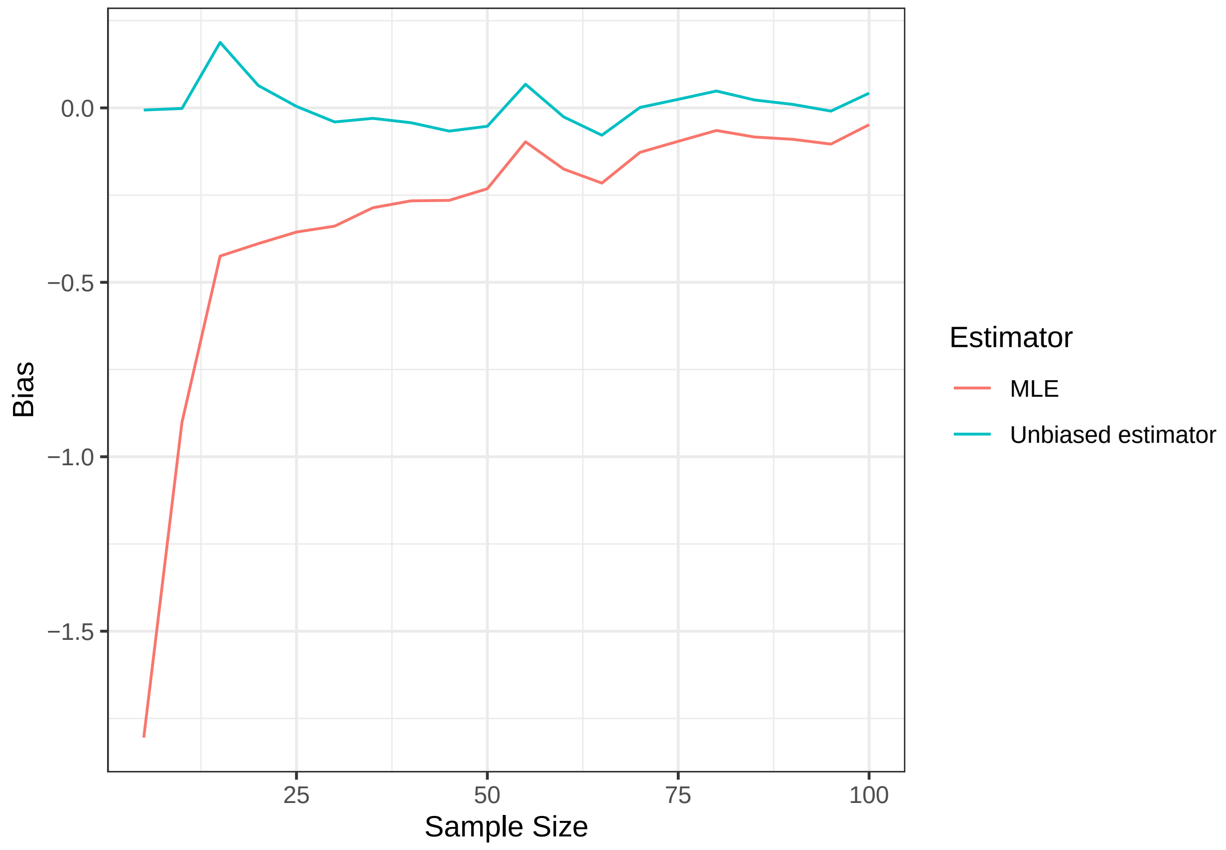
Conduct a simulation study which compares the bias of $\hat{V}_{\text{MLE}}$ as an estimate to the population variance σ_0^2 with the bias of $\hat{V}_U$ as an estimator for the population variance σ_0^2.

In your simulation study, you can consider different sample sizes ranging from 5 to 100 in increment of 5. For each sample size, conduct 1000 trials. For each trial, generate a samples $X_1, \cdots, X_n \sim \mathcal{N}(\mu_0, \sigma_0^2)$ with fixed parameters $\mu_0 = 1$ and $\sigma_0 = 3$, and then compute $\hat{V}_{\text{MLE}}$ and $\hat{V}_U$. Then create a plot which displays the bias of $\hat{V}_{\text{MLE}}$ and the bias of $\hat{V}_U$ as functions of the sample sizes.

Answer

```r
set.seed(0)
sample_size_seq <- seq(5, 100, 5)
num_trials_per_size <- 1000
mu_0 <- 1
sigma_0 <- 3

compute_V_mle <- function(x){ return( mean( (x-mean(x))^2 ) ) }
compute_V_U <- function(x){
  n <- length(x)
  return (compute_V_mle(x)*n/(n-1))
}

df <- crossing(sample_size=sample_size_seq, trials=seq(num_trials_per_size) ) %>%
  # create samples
  mutate(samples = map(sample_size, ~rnorm(.x, mean = mu_0, sd = sigma_0) ) ) %>%
  # compute V_mle
  mutate(V_mle = map_dbl(samples, compute_V_mle)) %>%
  # compute V_U
  mutate(V_U = map_dbl(samples, compute_V_U))

# compute bias
df_bias <- df %>% group_by(sample_size) %>%
  summarise(V_mle_bias=mean(V_mle)-sigma_0^2, V_U_bias=mean(V_U)-sigma_0^2)
```

```r
df_bias_longer <- df_bias %>%
  pivot_longer(c(V_mle_bias, V_U_bias), names_to = 'Estimator', values_to = 'Bias' ) %>%
  # change the names which will be displayed as Legend of the plot
  mutate(Estimator=case_when(Estimator=='V_mle_bias'~'MLE',
                             Estimator=='V_U_bias'~'Unbiased estimator'))

df_bias_longer %>% ggplot(aes(x=sample_size, y=Bias, color=Estimator)) + geom_line() +
  theme_bw() + xlab('Sample Size')
```



(Q2)

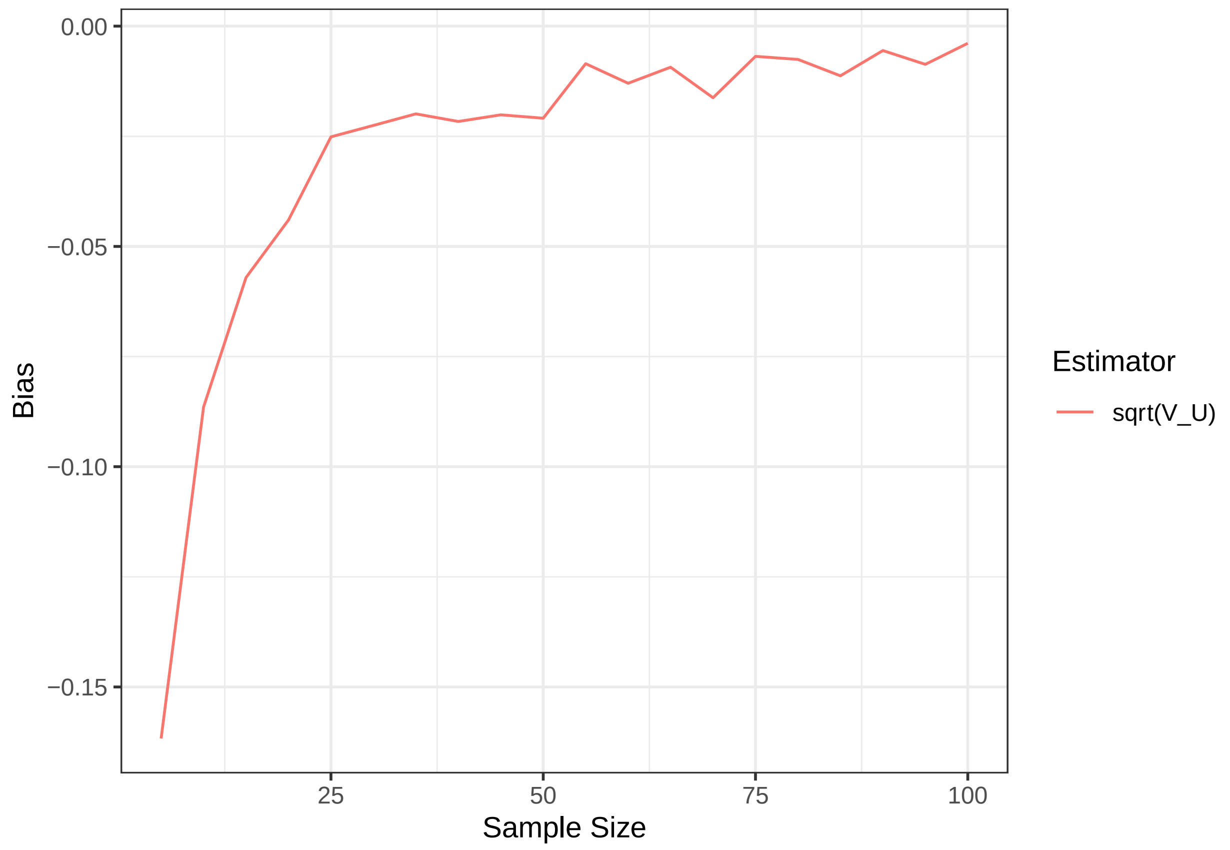
Is $\sqrt{\hat{V}_U} = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(X_i - \overline{X})^2}$ unbiased estimator for σ_0? You can conduct a simulation study (similar to in the last question) to answer this question.

Answer

```
set.seed(0)
sample_size_seq <- seq(5, 100, 5)
num_trials_per_size <- 10000
mu_0 <- 1
sigma_0 <- 3

compute_V_mle <- function(x){ return( mean( (x-mean(x))^2 ) ) }
compute_V_U <- function(x){
  n <- length(x)
  return (compute_V_mle(x)*n/(n-1))
}

df <- crossing(sample_size=sample_size_seq, trials=seq(num_trials_per_size) ) %>%
  # create samples
  mutate(samples = map(sample_size, ~rnorm(.x, mean = mu_0, sd = sigma_0) ) ) %>%
  # compute V_U_sqrt
  mutate(V_U_sqrt = map_dbl(samples, ~sqrt(compute_V_U(.x) )))

# compute bias
```

```r
df_bias <- df %>% group_by(sample_size) %>%
  summarise(V_U_sqrt_bias=mean(V_U_sqrt)-sigma_0)
```

```r
df_bias_longer <- df_bias %>%
  pivot_longer(c(V_U_sqrt_bias), names_to = 'Estimator', values_to = 'Bias' ) %>%
  # change the names which will be displayed as Legend of the plot
  mutate(Estimator=case_when(Estimator=='V_U_sqrt_bias'~'sqrt(V_U)'))

df_bias_longer %>% ggplot(aes(x=sample_size, y=Bias, color=Estimator)) + geom_line() +
  theme_bw() + xlab('Sample Size')
```



From the plot, we observe that $\mathbb{E}(\sqrt{\hat{V}_U})$ is consistently less than σ_0, for different sample sizes. Hence this suggest that $\sqrt{\hat{V}_U}$ is is biased.

Justification:

Since $\hat{V}_U$ is unbiased, we have

$$\mathbb{E}\left(\hat{V}_U\right) = \sigma_0^2$$

By Jensen's inequality

$$\mathbb{E}\left(\sqrt{\hat{V}_U}\right) < \sqrt{\mathbb{E}\left(\hat{V}_U\right)} = \sqrt{\sigma_0^2} = \sigma_0$$

Note that the equality in Jensen's inequality holds only when $\hat{V}_U$ is a constant, which is not the case here. So we have the strictly less than sign above.

(Q3) (optional) As an optional extra, give an analytic formula for the bias of $\hat{V}_{\text{MLE}}$ and $\hat{V}_U$.

Answer

Let's define $Z_i = (X_i - \mu_0)/\sigma_0$ for each $i = 1, \cdots, n$. Note that by independence we have $\mathbb{E}(Z_i Z_j) = 0$ if $i \neq j$ and $\mathbb{E}(Z_i^2) = 1$. Hence

$$\begin{aligned}
\frac{1}{\sigma_0^2}\mathbb{E}\left(\hat{V}_{\text{MLE}}\right) &= \frac{1}{\sigma_0^2}\mathbb{E}\left(\frac{1}{n}\sum_{i=1}^{n}\left(X_i - \frac{1}{n}\sum_{j=1}^{n}X_j\right)^2\right)\\
&= \frac{1}{\sigma_0^2}\mathbb{E}\left(\frac{1}{n}\sum_{i=1}^{n}\left(\sigma_0 Z_i - \frac{1}{n}\sum_{j=1}^{n}\sigma_0 Z_j\right)^2\right)\\
&= \mathbb{E}\left(\frac{1}{n}\sum_{i=1}^{n}\left(Z_i - \frac{1}{n}\sum_{j=1}^{n}Z_j\right)^2\right)\\
&= \frac{1}{n}\mathbb{E}\left(\sum_{i=1}^{n}\left(\frac{n-1}{n}Z_i - \frac{1}{n}\sum_{j\neq i}Z_j\right)^2\right)\\
&= \frac{1}{n}\left(\sum_{i=1}^{n}\left(\left(\frac{n-1}{n}\right)^2\mathbb{E}\big(Z_i^2\big) - \frac{1}{n^2}\sum_{j\neq i}\mathbb{E}\big(Z_j^2\big)\right)\right)\\
&= \frac{n-1}{n}
\end{aligned}$$

Therefore, $\mathbb{E}\left(\hat{V}_{\text{MLE}}\right) = \frac{n-1}{n}\sigma_0^2$, and $\text{Bias}(\hat{V}_{\text{MLE}}) = -\frac{\sigma_0^2}{n}$ and $\text{Bias}(\hat{V}_U) = 0$.

1.3 Maximum likelihood estimation with the Poisson distribution

In this question we shall consider the topic of maximum likelihood estimation for an independent and identically distributed sample from a Poisson random variable. Recall that Poisson random variables are a family of discrete random variables with distributions supported on $\mathbb{N}_0 := \{0, 1, 2, \cdots, \}$

Poisson random variables are frequently used to model the number of events which occur at a constant rate in situations where the occurrences of individual events are independent. For example, we might use the Poisson distribution to model the number of mutations of a given strand of DNA per time unit, or the number of customers who arrive at the store over the course of a day. A classic example of statistical modelling based on a Poisson distribution is due to the statistician Ladislaus Josephovich Bortkiewicz. Bortkiewicz used the Poisson distribution to model the number of fatalities due to horse-kick per year for each group of cavalry. We shall apply maximum likelihood estimation to Bortkiewicz's data. First, let's explore maximum likelihood estimation for Poisson random variables.

A Poisson random variable has a probability mass function $p_\lambda : \mathbb{R} \to (0, \infty)$ with a single parameter $\lambda > 0$. The probability mass function $p_\lambda : \mathbb{R} \to (0, \infty)$ is defined for $x \in \mathbb{R}$ by

$$p_\lambda = \begin{cases} \frac{\lambda^x e^{-\lambda}}{x!} & \text{for} \quad x \in \mathbb{N}_0, \\ 0 & \text{for} \quad x \neq \mathbb{N}_0. \end{cases}$$

Suppose that you have a sample of independent and identically distributed random variables $X_1, \cdots, X_n \sim p_{\lambda_0}$, i.e., $X_1, \cdots, X_n$ are independent and each has probability mass function p_{λ_0}.

(Q1)

Show that for a sample $X_1, \cdots, X_n$, the likelihood function $l : (0, \infty) \to (0, \infty)$ is given by

$$l(\lambda) = e^{-n\lambda} \cdot \lambda^{n \cdot \overline{X}} \cdot \left(\prod_{i=1}^{n} \frac{1}{X_i!} \right),$$

where $\overline{X} = \frac{1}{n} \sum_{i=1}^{n} X_i$ is the sample mean.

Then derive a formula for the derivative of the log-likelihood $\frac{\partial}{\partial \lambda} \log l(\lambda)$.

Answer

The likelihood function is

$$l(\lambda) = \prod_{i=1}^{n} p_\lambda(X_i) = e^{-n\lambda} \cdot \lambda^{n \cdot \overline{X}} \cdot \left(\prod_{i=1}^{n} \frac{1}{X_i!} \right)$$

The log-likelihood function is

$$\log l(\lambda) = -n\lambda + n\overline{X} \log(\lambda) + \log(\prod_{i=1}^{n} \frac{1}{X_i!})$$

Therefore, the derivative

$$\frac{\partial}{\partial \lambda} \log l(\lambda) = -n + \frac{n\overline{X}}{\lambda}$$

(Q2)

Show that $\lambda \to \log l(\lambda)$ reaches its maximum at the single point at which $\lambda = \overline{X}$. Hence, the maximum likelihood estimate for the true parameter λ_0 is $\hat{\lambda}_{\text{MLE}} = \overline{X}$.

Answer

The derivative $\frac{\partial}{\partial \lambda} \log l(\lambda)$ is bigger than zero when $\lambda < \overline{X}$, and less than zero when $\lambda > \overline{X}$. So $\lambda \to \log l(\lambda)$ reaches its maximum at the single point at which $\lambda = \overline{X}$. Hence, the maximum likelihood estimate for the true parameter λ_0 is $\hat{\lambda}_{\text{MLE}} = \overline{X}$.

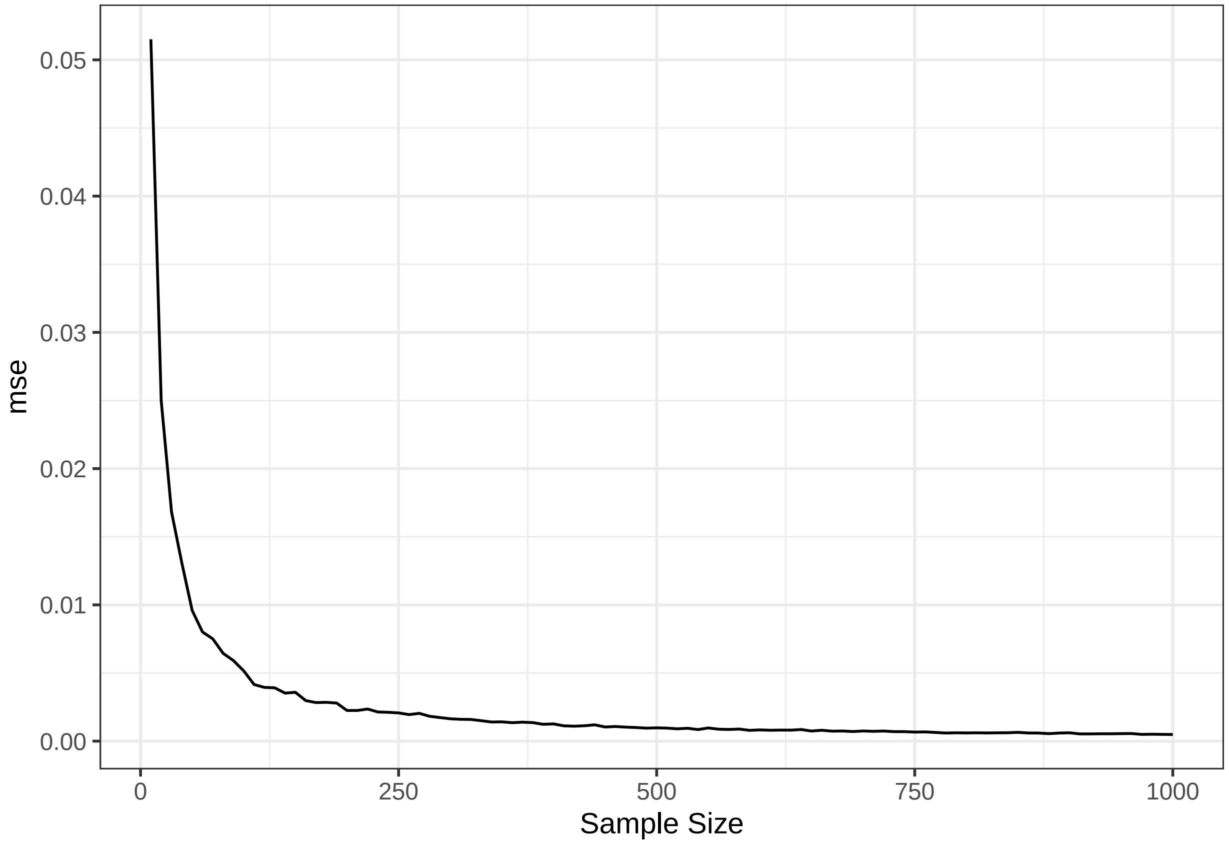
(Q3)

Now conduct a simulation experiment which explores the behaviour of $\hat{\lambda}_{\text{MLE}}$ on simulated data. You may wish to consider a setting in which $\lambda_0 = 0.5$ and generate a plot of the mean squared error as a function of the sample size. To generate samples from Poisson distribution, you can consider the `rpois()` function. There is no specific requirement on the plot or the range of sample sizes etc, but you should make sure your results clearly display your conclusions.

Answer

```r
lambda_0 = 0.5
num_trials_per_sample_size=1000

df = crossing(sample_size=seq(10, 1000, 10), trials=seq(num_trials_per_sample_size)) %>%
  # create samples
  mutate(samples=map(sample_size, ~rpois(.x, lambda_0)) ) %>%
  # compute MLE
  mutate(lambda_mle = map_dbl(samples, ~mean(.x)))

df_mse <- df %>% group_by(sample_size) %>%
  summarise(mse=mean( (lambda_mle-lambda_0)^2 ))
```

```r
ggplot() + geom_line(data=df_mse, aes(x=sample_size, y=mse)) +
  theme_bw() + xlab('Sample Size')
```



(Q4)

Now that we have explored maximum likelihood estimation with a Poisson distribution for simulated data we shall return to Poisson modelling with real data. Let's take a look at the famous horse-kick fatality data set explored by Ladislaus Josephovich Bortkiewicz. A csv file containing this data is available within Blackboard. The file name is `VonBortkiewicz.csv`.

Download the csv file and load the file into an R data frame. You may wish to use the read.csv() function.

The count data for horse fatalities per year, per cavalry corps, are given in the "fatalities" column. Model the values in this column as independent random variables $X_1, \cdots, X_n$ from a Poisson distribution with parameter λ_0 and compute the maximum likelihood estimate $\hat{\lambda}_{\text{MLE}}$ for λ_0.

Use your fitted Poisson model to give an estimate for the probability that a single cavalry corps has no fatalities due to horse kicks in a single year. You may want to use the `dpois` function.

Answer

```r
real_data <- read.csv("VonBortkiewicz.csv")
# note: if the .csv file is not in your current folder, then
# you will need to specify the folder where this file is in.

# MLE:
lambda_mle <- mean(real_data$fatalities )
print(lambda_mle)
```

```
## [1] 0.7
```

```
predict_prob_no_fatalities <- dpois(0, lambda_mle)
print(predict_prob_no_fatalities)
```

```
## [1] 0.4965853
```

```
print(mean(real_data$fatalities==0)) # for comparison
```

```
## [1] 0.5142857
```

(Q5) (optional)

As an optional extra give a formula for $\mathcal{I}(\lambda) := -\mathbb{E}\left(\frac{\partial^2}{\partial\lambda^2}\log p_\lambda(X)\right)$ where $X \sim p_\lambda$ is a Poisson random variable with rate λ. Next generate a simulation involving random samples of size 1000 from a Poisson random variable with parameter $\lambda = 0.5$. Give a kernel density plot of $\sqrt{n\mathcal{I}(\lambda_0)}(\hat{\lambda}_{\text{MLE}} - \lambda_0)$.

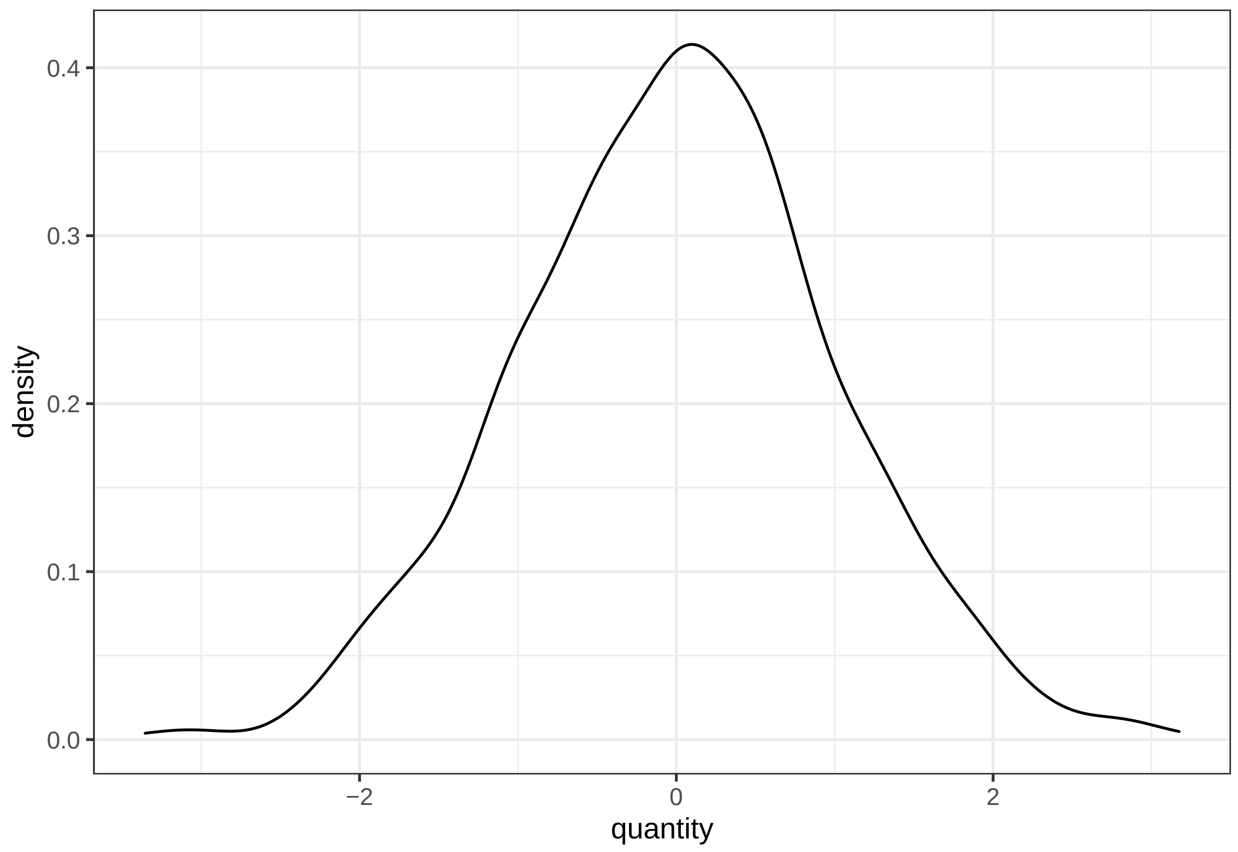
Answer

$$\frac{\partial}{\partial\lambda}\log p_\lambda(X) = -n + \frac{X}{\lambda}$$

$$\frac{\partial^2}{\partial\lambda^2}\log p_\lambda(X) = -\frac{X}{\lambda^2}$$

$$\begin{aligned}\mathcal{I}(\lambda) &:= -\mathbb{E}\left(\frac{\partial^2}{\partial\lambda^2}\log p_\lambda(X)\right)\\ &= -\mathbb{E}\left(-\frac{X}{\lambda^2}\right)\\ &= \frac{1}{\lambda}\end{aligned}$$

```
set.seed(0)
lambda_0 = 0.5
sample_size <- 1000

df <- data.frame(trials=seq(1000)) %>%
  # create samples
  mutate(sample= map(trials, ~rpois(sample_size, lambda_0))) %>%
  # compute MLE
  mutate(MLE = map_dbl(sample, mean) ) %>%
  # the quality of interest
  mutate(quantity = sqrt(sample_size/lambda_0)*(MLE-lambda_0) )

# create kernel density plot
ggplot(df, aes(x=quantity) ) + geom_density() +
  theme_bw()
```



The kernel density looks similar to the density of a standard Gaussian random variable.

1.4 Maximum likelihood estimation for the exponential distribution

Recall from our last assignment that given a positive real number $\lambda > 0$, an exponential random variable X with parameter λ is a continuous random variable with density $p_\lambda : \mathbb{R} \to (0, \infty)$ define by

$$p_\lambda(x) = \begin{cases} 0 & \text{if} \quad x < 0 \\ \lambda e^{-\lambda x} & \text{if} \quad x \geq 0. \end{cases}$$

(Q1)

Suppose that $X_1, \cdots, X_n$ is an i.i.d sample from the exponential distribution with an unknown parameter $\lambda_0 > 0$. What is the maximum likelihood estimate for λ_0?

Answer

The log-likelihood function is

$$\log l(\lambda) = n \log \lambda - \lambda \sum_{i=1}^{n} X_i$$

$$\frac{\partial}{\partial \lambda} \log l(\lambda) = \frac{n}{\lambda} - \sum_{i=1}^{n} X_i$$

If $\lambda < 1/\overline{X} := \frac{1}{n} \sum_{i=1}^{n} X_i$, then $\frac{\partial}{\partial \lambda} \log l(\lambda) > 0$.

If $\lambda > 1/\overline{X} := \frac{1}{n}\sum_{i=1}^{n} X_i$, then $\frac{\partial}{\partial\lambda}\log l(\lambda) < 0$.

So the maximum likelihood estimate for λ_0 is $\hat{\lambda}_{\text{MLE}} = 1/\overline{X}$.

(Q2)

We shall now use the exponential distribution to model the differences in purchase times between customers at a large supermarket. In Blackboard you will find the "CustomerPurchase" csv file. Download the "CustomerPurchase" csv file and load the file into an R data frame. You may wish to use the read.csv() function. The first column is the purchase time given in seconds since the store opens.

Add a new column in your data frame called "`time_diffs`" which gives the time in seconds until the next customer's purchase. That is, letting $Y_1, Y_2, \cdots, Y_{n+1}$ denote the sequence of arrival times in seconds, the `time_diffs` column contains $X_1, \cdots, X_n$ where $X_i = Y_{i+1} - Y_i$ for each $i = 1, \cdots, n$. You can let last row of "`time_diffs`" be NA (missing value). You may want to use the `lead()` function.

Answer

```r
CustomerPurchase <- read.csv("CustomerPurchase.csv")
# note: if the .csv file is not in your current folder, then
# you will need to specify the folder where this file is in.

CustomerPurchase <- CustomerPurchase %>%
  mutate(time_diff=lead(Time)-Time)

head(CustomerPurchase)
```

```
##   Time Purchase time_diff
## 1  564     3.25         7
## 2  571   504.85         7
## 3  578     7.60        22
## 4  600    43.45       145
## 5  745     9.30        61
## 6  806   352.80        27
```

(Q3)

Model the sequence of differences in purchase times $X_1, \cdots, X_n$ as independent and identically distributed exponential random variables. Compute the maximum likelihood estimate of the rate parameter $\hat{\lambda}_{\text{MLE}}$.

Answer

```r
lambda_mle = 1/mean(CustomerPurchase$time_diff, na.rm=TRUE)
lambda_mle
```

```
## [1] 0.02007792
```

(Q4)

Use your fitted exponential model to give an estimate of the probability of an arrival time in excess of one minute. You may wish to make use of the `pexp()` function.

Answer

```r
prob_excess_one_minute <- 1 - pexp(60, rate=lambda_mle)
prob_excess_one_minute
```

```
## [1] 0.2997893
```

2. Confidence intervals

2.1 Student's t-confidence intervals

In this problem we will discuss a parametric approach to obtaining confidence intervals based upon Student's t-distribution. In the code below "`adelie_flippers`" is a vector containing the flipper lengths of a sample of Adelie penguins. The following code computes confidence intervals based on "`adelie_flippers`" for the population mean of the flipper lengths for Adelie penguins using the Student's t-distribution method.

```r
alpha <- 0.05
sample_size <- length(adelie_flippers) # adelie_flippers is a given vector
sample_mean <- mean(adelie_flippers)
sample_sd <- sd(adelie_flippers)
t <- qt(1-alpha/2,df=sample_size-1)
# confidence interval
confidence_interval_l <- sample_mean-t*sample_sd/sqrt(sample_size)
confidence_interval_u <- sample_mean+t*sample_sd/sqrt(sample_size)
confidence_interval <- c(confidence_interval_l,confidence_interval_u)
confidence_interval
```

(Q1)

What would happen to the width of my confidence interval if the sample mean were higher? What would happen to the width of my confidence interval if the sample standard deviation were higher? What would happen to the width of my confidence interval if the sample size were larger (keeping the sample standard deviation the same)?

Answer

1) If the sample mean were higher, then the width of the confidence interval does not change
2) If the sample standard deviation were higher, then the width increase
3) If the sample size were larger and the sample standard deviation remained unchanged, then the width of the confidence interval is smaller

(Q2)

Use your data wrangling skills to extract a vector consisting of the weights of all the Red-Tailed hawks from the "Hawks" data set, with any missing values removed.

Now use the Student's t method to compute 99%-level confidence intervals for the population mean of the weights for the red-tailed hawks. Note that opting for confidence intervals with a confidence level of 99%, rather than a confidence level of 95%, requires a modified value of α.

Answers

```r
weights <- Hawks %>% filter(Species=="RT" ) %>%
  select(Weight) %>%
  filter(complete.cases(Weight)) %>%
  # removing NAs; you can also use discard(is.na) after calling pull()
  pull()
```

```r
alpha <- 0.01
sample_size <- length(weights) # adelie_flippers is a given vector
sample_mean <- mean(weights)
sample_sd <- sd(weights)
t <- qt(1-alpha/2,df=sample_size-1)
# confidence interval
```

```
confidence_interval_l <- sample_mean-t*sample_sd/sqrt(sample_size)
confidence_interval_u <- sample_mean+t*sample_sd/sqrt(sample_size)
confidence_interval <- c(confidence_interval_l,confidence_interval_u)
confidence_interval
```

```
## [1] 1073.984 1114.877
```

(Q3)

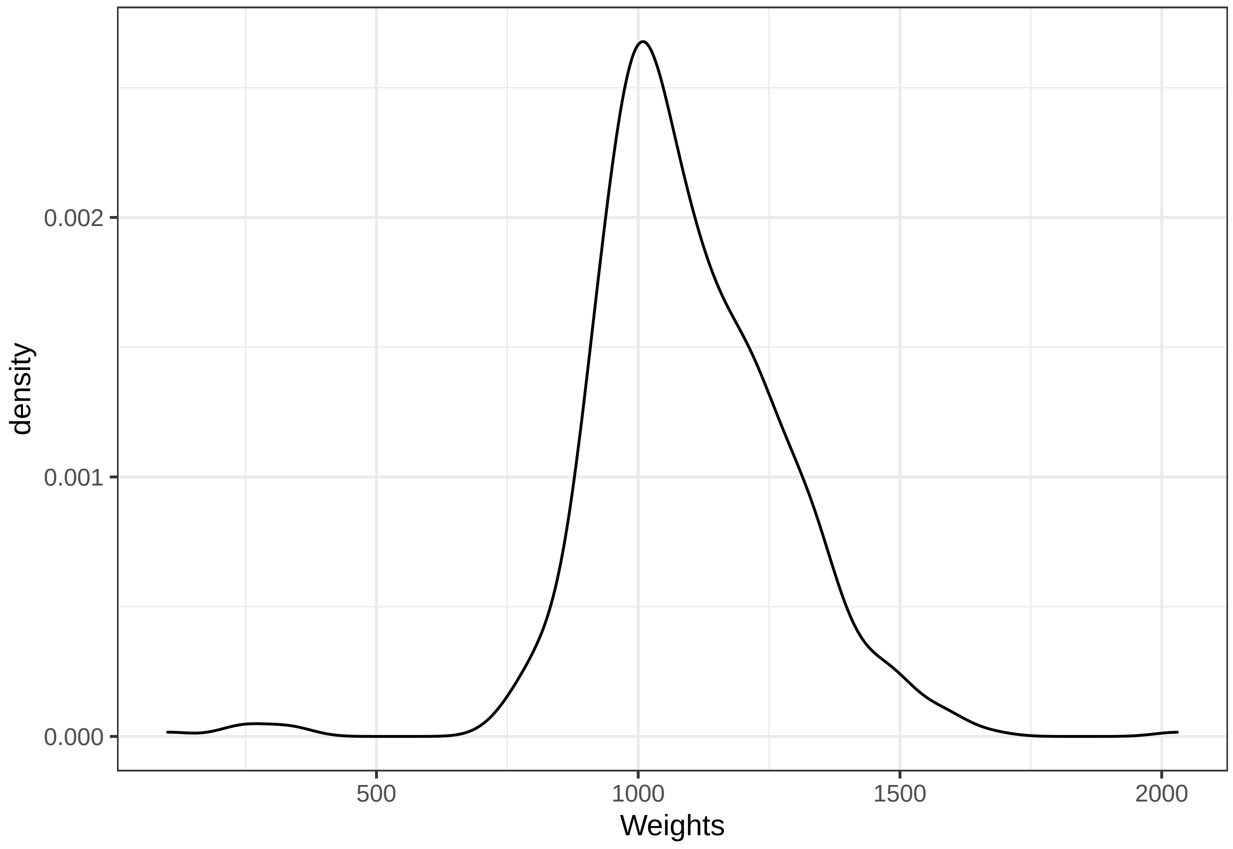
What assumptions are made to derive confidence intervals based on Student's t-distribution? Check if these assumptions are justified using a kernel density plot with the `geom_density()` function and using a `QQ`-plot with the `stat_qq`() function.

Answer

The student's t confidence intervals require the data to be Gaussian or approximately Gaussian.

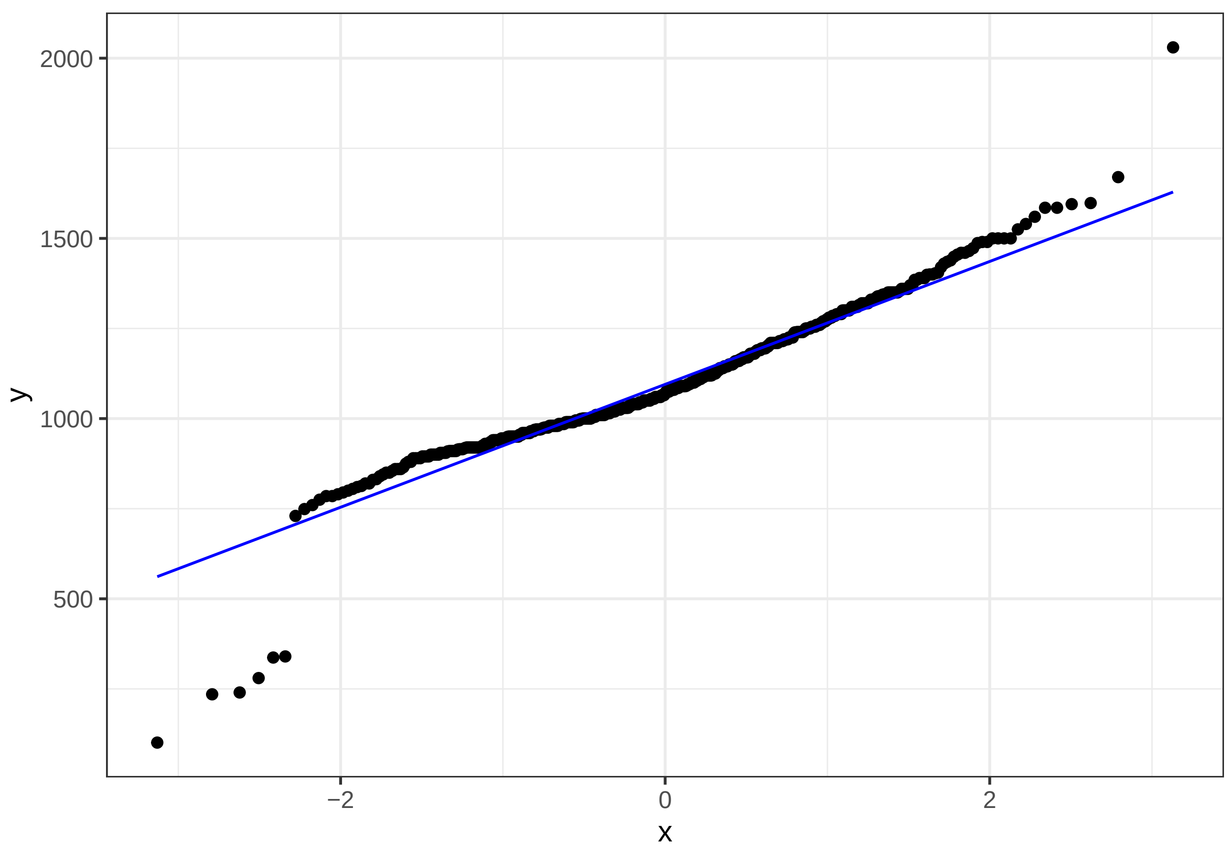
First we create a kernel density plot. This reveals that the distribution is unimodal but has slightly heavier tails than a Gaussian.

```
ggplot(data=data.frame(Weights=weights), aes(x=Weights)) + geom_density() +
  theme_bw()
```



Next we generate a QQ plot. Here we again notice the heavy tails visible within the QQ-plot.

```
ggplot(data=data.frame(Weights=weights), aes(sample=Weights)) + stat_qq() +
  theme_bw() + stat_qq_line(color="blue")
```



However, the sample size is relatively large, so we are reasonably content to use Student's t based intervals.

```r
Hawks%>%filter(Species=="RT")%>%nrow()
```

```
## [1] 577
```

2.2 Investigating coverage for Student's t intervals

In this question we shall assume that we have access to a sample $X_1, \cdots, X_n \sim \mathcal{N}(\mu_0, \sigma_0^2)$ consisting of i.i.d. Gaussian data. We are interested in determining the value of the unknown population mean μ_0 based on the sample $X_1, \cdots, X_n$.

Suppose we wish to compute confidence intervals for μ_0 with confidence level $(1-\alpha) \times 100\%$ for some $\alpha \in (0,1)$. For example, we could have $\alpha = 0.05$, in which cases we wish to compute confidence intervals with confidence level 95%.

Let $\overline{X} := \frac{1}{n}\sum_{i=1}^n X_i$ be the sample mean and $S := \sqrt{\frac{1}{n-1}\sum_{i=1}^n (X_i - \overline{X})^2}$ be the sample standard deviation. In addition, let $t_{\alpha/2,n-1}$ be the $(1-\frac{\alpha}{2})$-quantile of the Student's t-distribution with $n-1$ degrees of freedom.

The Student's t confidence interval for μ_0 is given by $(L_\alpha(X_1, \cdots, X_n), R_\alpha(X_1, \cdots, X_n))$ defined by

$$L_\alpha(X_1, \cdots, X_n) := \overline{X} - \frac{t_{\alpha/2,n-1}}{\sqrt{n}} \cdot S$$

$$R_\alpha(X_1, \cdots, X_n) := \overline{X} + \frac{t_{\alpha/2,n-1}}{\sqrt{n}} \cdot S$$

The following code generates a function `student_t_confidence_interval`, which takes as input a sample $X_1, \cdots, X_n$ given as a vector along with a confidence level $\gamma = 1 - \alpha$ and outputs a tuple containing $(L_\alpha(X_1, \cdots, X_n), R_\alpha(X_1, \cdots, X_n))$

```r
student_t_confidence_interval<-function(sample,confidence_level){
  sample<-sample[!is.na(sample)] # remove any missing values
  n<-length(sample) # compute sample size
  mu_est<-mean(sample) # compute sample mean
  sig_est<-sd(sample) # compute sample sd
  alpha = 1-confidence_level # alpha from gamma

  t<-qt(1-alpha/2,df=n-1) # get student t quantile
  l=mu_est-(t/sqrt(n))*sig_est # lower
  u=mu_est+(t/sqrt(n))*sig_est # upper
return(c(l,u))
}
```

Check that you understand this function and implement it for yourself.

The key property of a confidence interval for μ_0 at the confidence level of $(1-\alpha)\times 100\%$ is the following coverage property:

$$\mathbb{P}\{(L_\alpha(X_1,\cdots,X_n)\leq\mu_0\leq R_\alpha(X_1,\cdots,X_n))\}\geq 1-\alpha.$$

This is known as a coverage property since it tells us that the confidence interval covers μ_0 with probability $1-\alpha$. The following simulation checks this property with $\mu_0=1,\sigma_0=3$ and a confidence level of 95% i.e. $\gamma=0.95$.

```r
num_trials <- 100000
sample_size <- 30
mu_0 <- 1
sigma_0 <- 3
alpha <- 0.05
set.seed(0) # set random seed for reproducibility

single_alpha_coverage_simulation_df <- data.frame(trial=seq(num_trials)) %>%
  # generate random Gaussian samples:
  mutate(sample=map(.x=trial,.f=~rnorm(n=sample_size,mean=mu_0,sd=sigma_0))) %>%
  # generate confidence intervals:
  mutate(ci_interval=map(.x=sample, .f=~student_t_confidence_interval(.x,1-alpha)))%>%
  # check if interval covers mu_0:
  mutate(cover=map_lgl(.x=ci_interval, .f=~((min(.x)<=mu_0)&(max(.x)>=mu_0))))%>%
  # compute interval length:
  mutate(ci_length=map_dbl(.x=ci_interval, .f=~(max(.x)-min(.x))))

# estimate of coverage probability:
single_alpha_coverage_simulation_df %>%
  pull(cover) %>%
  mean()
```

```
## [1] 0.95003
```

(Q1)

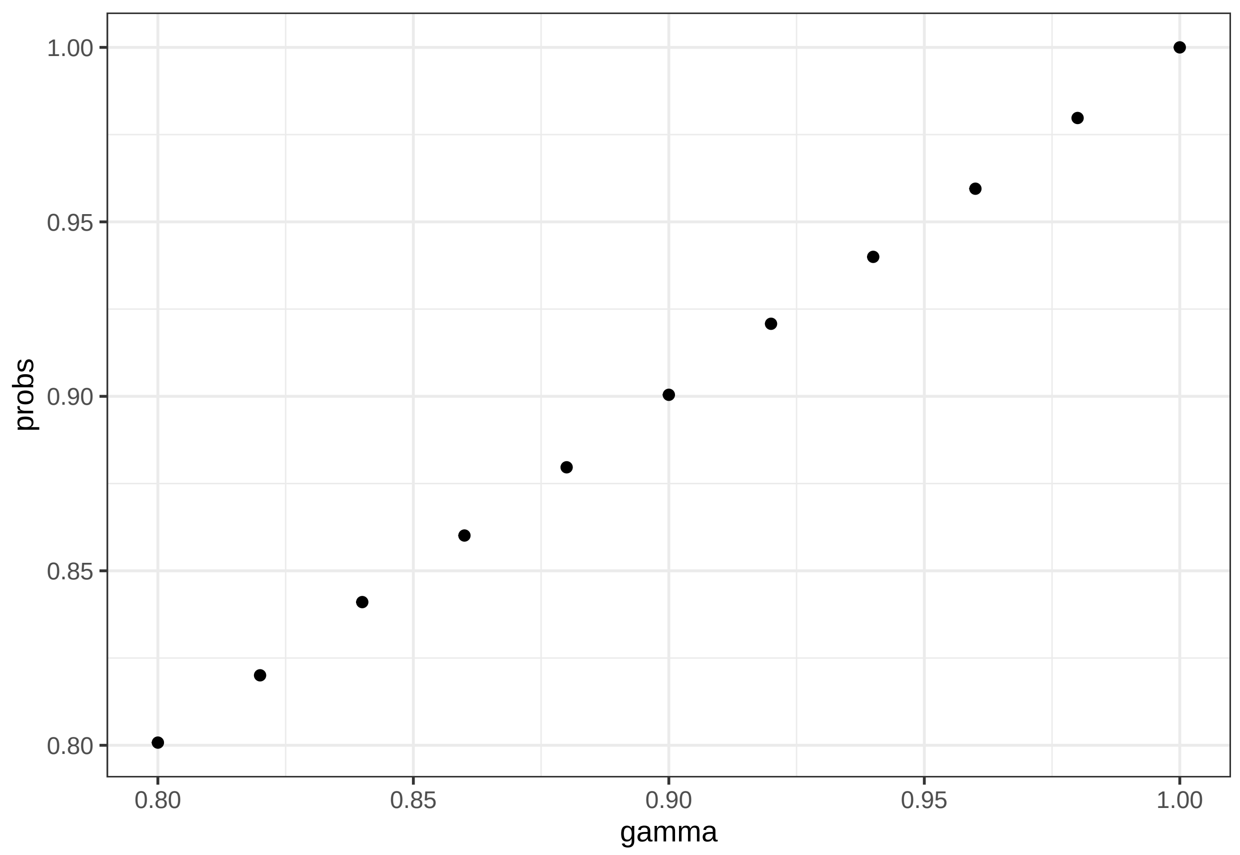
Check that you understand the above code. Now modify the above code to conduct a simulation experiment to investigate how $\mathbb{P}\{(L_\alpha(X_1,\cdots,X_n)\leq\mu_0\leq R_\alpha(X_1,\cdots,X_n))\}$ varies as a function of the confidence level $\gamma=1-\alpha$.

Answer

```r
set.seed(0) # set random seed for reproducibility
```

```r
probs_CI_contains_mu <- function(gamma){
num_trials <- 100000
sample_size <- 30
mu_0 <- 1
sigma_0 <- 3
alpha <- 1-gamma

single_alpha_coverage_simulation_df <- data.frame(trial=seq(num_trials)) %>%
  # generate random Gaussian samples:
  mutate(sample=map(.x=trial,.f=~rnorm(n=sample_size,mean=mu_0,sd=sigma_0))) %>%
  # generate confidence intervals:
  mutate(ci_interval=map(.x=sample, .f=~student_t_confidence_interval(.x,1-alpha)))%>%
  # check if interval covers mu_0:
  mutate(cover=map_lgl(.x=ci_interval, .f=~((min(.x)<=mu_0)&(max(.x)>=mu_0))))%>%
  # compute interval length:
  mutate(ci_length=map_dbl(.x=ci_interval, .f=~(max(.x)-min(.x))))

# estimate of coverage probability:
single_alpha_coverage_simulation_df %>%
  pull(cover) %>%
  mean()
}

df <- data.frame(gamma=seq(0.8,1,0.02)) %>%
  mutate(probs=map_dbl(gamma, probs_CI_contains_mu))
```

```r
ggplot() + geom_point(data=df, aes(x=gamma, y=probs)) +
  theme_bw()
```



The results suggest that $\mathbb{P}\{(L_\alpha(X_1,\cdots,X_n)\le\mu_0\le R_\alpha(X_1,\cdots,X_n))\}$ is very close to γ. In fact $\mathbb{P}\{(L_\alpha(X_1,\cdots,X_n)\le\mu_0\le R_\alpha(X_1,\cdots,X_n))\}=\gamma$.

(Q2)

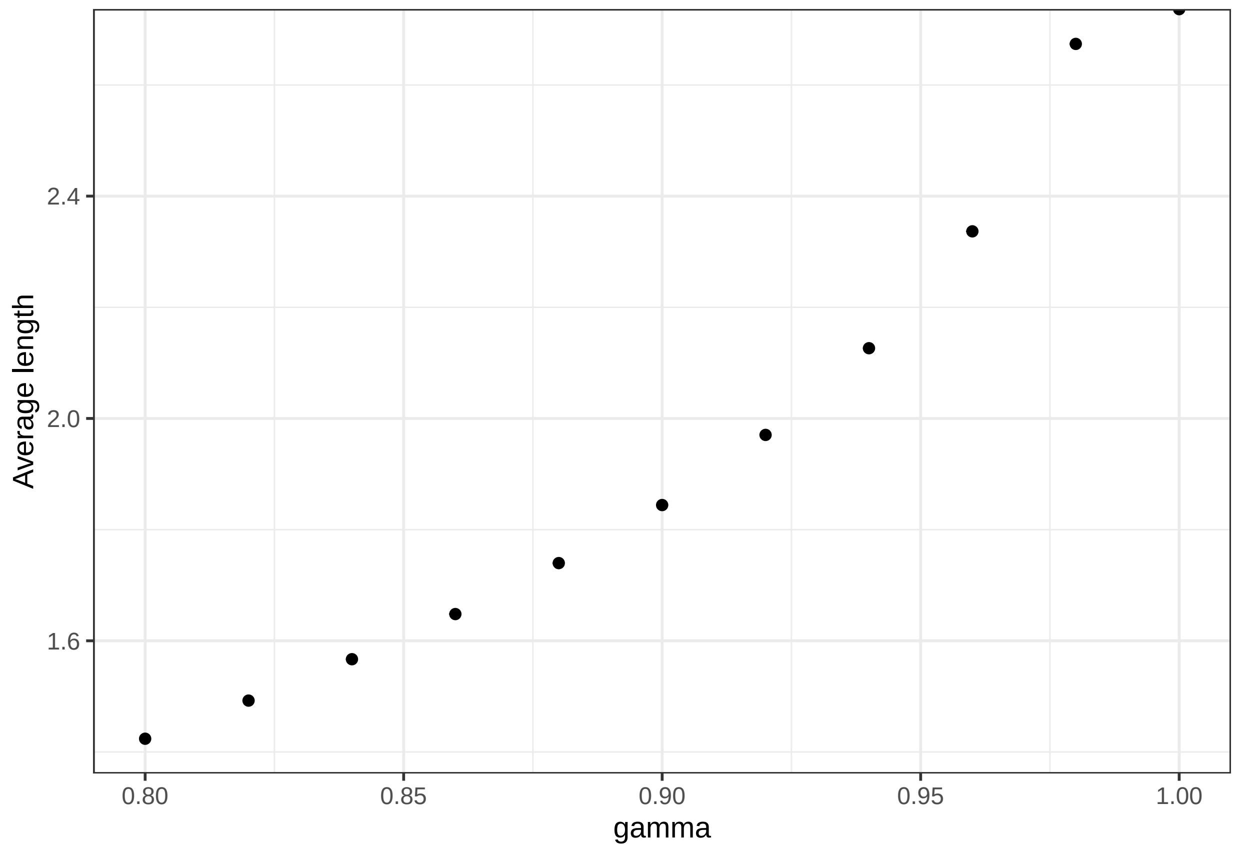
How does the average length $\mathbb{E}(R_\alpha(X_1,\cdots,X_n)-R_\alpha(X_1,\cdots,X_n))$ vary as a function of the confidence level $\gamma=1-\alpha$? You may want to conduct a simulation study to answer this question, similar to the one in the last question.

Answer

```
set.seed(0) # set random seed for reproducibility

compute_CI_width <- function(gamma){
num_trials <- 100000
sample_size <- 30
mu_0 <- 1
sigma_0 <- 3
alpha <- 1-gamma

single_alpha_coverage_simulation_df <- data.frame(trial=seq(num_trials)) %>%
  # generate random Gaussian samples:
  mutate(sample=map(.x=trial,.f=~rnorm(n=sample_size,mean=mu_0,sd=sigma_0))) %>%
  # generate confidence intervals:
  mutate(ci_interval=map(.x=sample, .f=~student_t_confidence_interval(.x,1-alpha)))%>%
  # check if interval covers mu_0:
  mutate(cover=map_lgl(.x=ci_interval, .f=~((min(.x)<=mu_0)&(max(.x)>=mu_0))))%>%
  # compute interval length:
```

```r
  mutate(ci_length=map_dbl(.x=ci_interval, .f=~(max(.x)-min(.x))))

# estimate of coverage probability:
single_alpha_coverage_simulation_df %>%
  pull(ci_length) %>%
  mean()
}

df <- data.frame(gamma=seq(0.8,1,0.02)) %>%
  mutate(averagewidth=map_dbl(gamma, compute_CI_width))
```

```r
ggplot() + geom_point(data=df, aes(x=gamma, y=averagewidth)) +
  theme_bw() + ylab('Average length')
```



The average length increases as the confidence level γ increases.

3. One sample hypothesis testing

3.1 One sample t-test on penguins data

In this question, we will apply one sample t-test on the population mean of the Adelie penguin's bill lengths.

(Q1)

Begin by loading the "Palmer penguins" library. Next extract a vector called "`bill_adelie`" consisting of the bill lengths of the Adelie penguins belonging to the Adelie species.

Carry out a statistical hypothesis test to test the hypothesis that the population mean of the Adelie penguin's bill lengths is 40 mm. Use a significance level of 0.01. You can use the `t.test()` function. What assumptions are required for this hypothesis test?

Answer

```
library(palmerpenguins)
```

```
bill_adelie <- penguins %>% filter(species=='Adelie') %>%
  select(bill_length_mm) %>% pull() %>% discard(is.na)
```

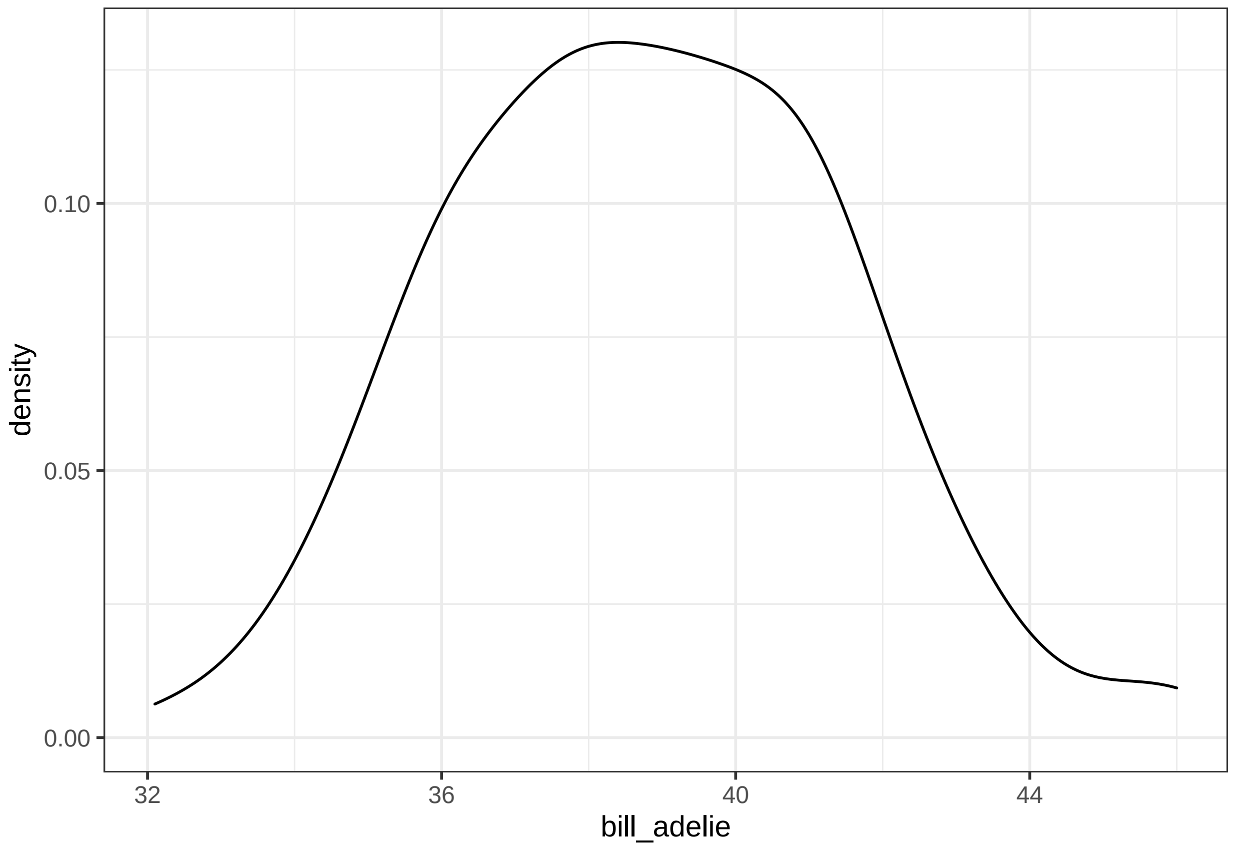
```
t.test(bill_adelie, mu=40, alternative = 'two.sided')
```

```
##
##  One Sample t-test
##
## data:  bill_adelie
## t = -5.5762, df = 150, p-value = 1.114e-07
## alternative hypothesis: true mean is not equal to 40
## 95 percent confidence interval:
##  38.36312 39.21966
## sample estimates:
## mean of x
##  38.79139
```

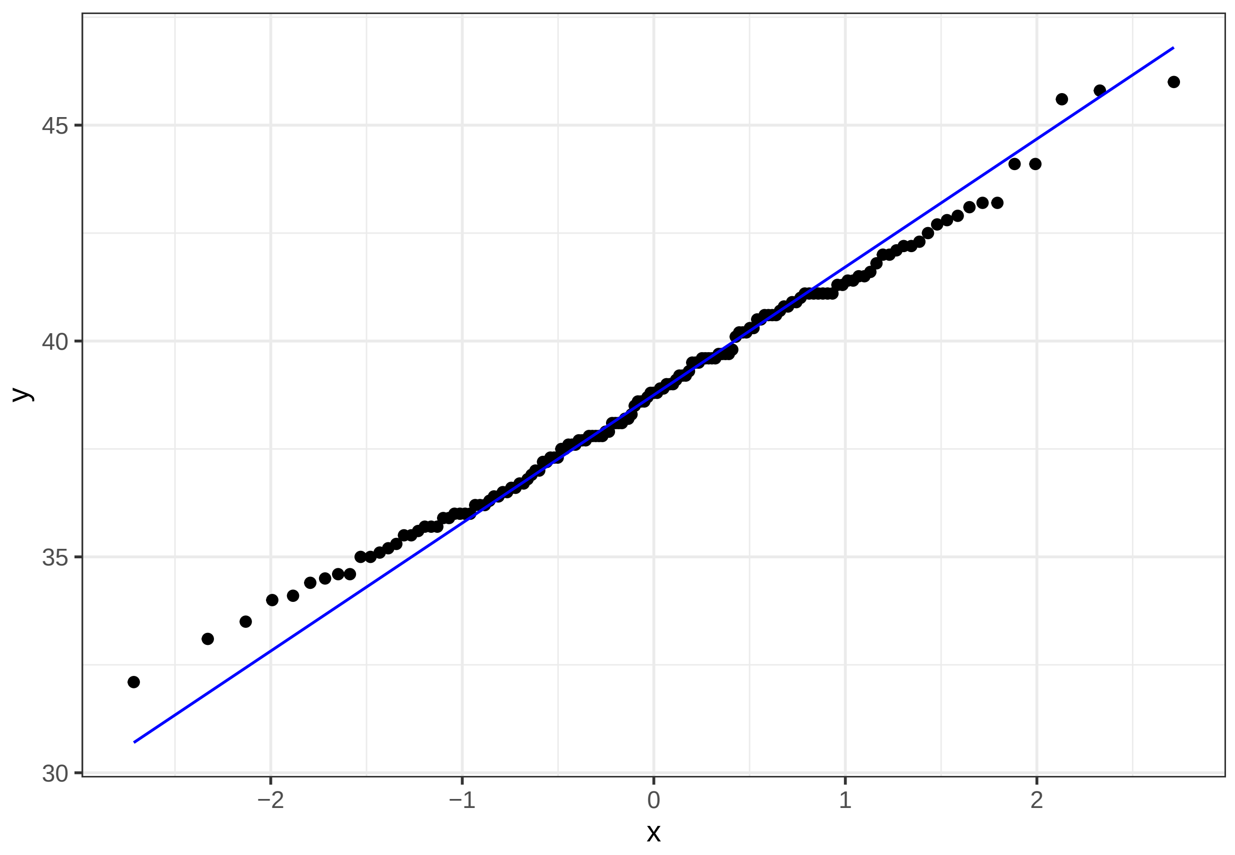
To do the t-test, your data must be Gaussian or approximately Gaussian.

To check this modelling assumption, we can use the density plot and QQ-plot to investigate the distribution of our data:

```
ggplot(data=data.frame(bill_adelie=bill_adelie), aes(x=bill_adelie)) + geom_density() +
  theme_bw()
```



```r
ggplot(data=data.frame(bill_adelie=bill_adelie), aes(sample=bill_adelie)) + stat_qq() +
  theme_bw() + stat_qq_line(color="blue")
```



3.2 Implementing a one-sample t-test

(Q1)

Implement a function that carries out a two-sided one-sample t-test. Your function should take in two arguments

1) a vector `x` corresponding to a sample $X_1, \cdots, X_n \in \mathcal{N}(\mu, \sigma^2)$ and
2) the value μ_0 corresponding to a null hypothesis of $\mu = \mu_0$.

The output of your function should be the corresponding p-value of the test.

You can test your implementation by confirming your function gives the same p-value as the `t.test()` function for the example in question 3.1(Q1) of the assignment.

Answer

```
my_t_test <- function(x, mu0){
  sample_size <- length(x)
  sample_mean <- mean(x)
  sample_sd <- sd(x)

  test_statistic <- (sample_mean - mu0)/(sample_sd/sqrt(sample_size))
  p_value = 2*(1-pt(abs(test_statistic), df=sample_size-1))

  return(p_value)
}
```

```r
my_t_test(bill_adelie, mu=38.5)
```

```
## [1] 0.1808501
```

```r
t.test(bill_adelie, mu=38.5, alternative = 'two.sided')
```

```
##
##  One Sample t-test
##
## data:  bill_adelie
## t = 1.3444, df = 150, p-value = 0.1809
## alternative hypothesis: true mean is not equal to 38.5
## 95 percent confidence interval:
##  38.36312 39.21966
## sample estimates:
## mean of x
##  38.79139
```

The two functions give the same p-value.