Assignment 6 solutions

EMATM0061: Statistical Computing and Empirical Methods, TB1, 2024

Introduction

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 load the tidyverse package:

```r
library(tidyverse)
```

1. The Gaussian distribution

Gaussian random variables are an important family of continuous random variables. In this assignment, we will first explore several properties of the Gaussian distribution.

Use the help function to look up the following four functions: "dnorm()", "pnorm()", "qnorm()" and "rnorm()".

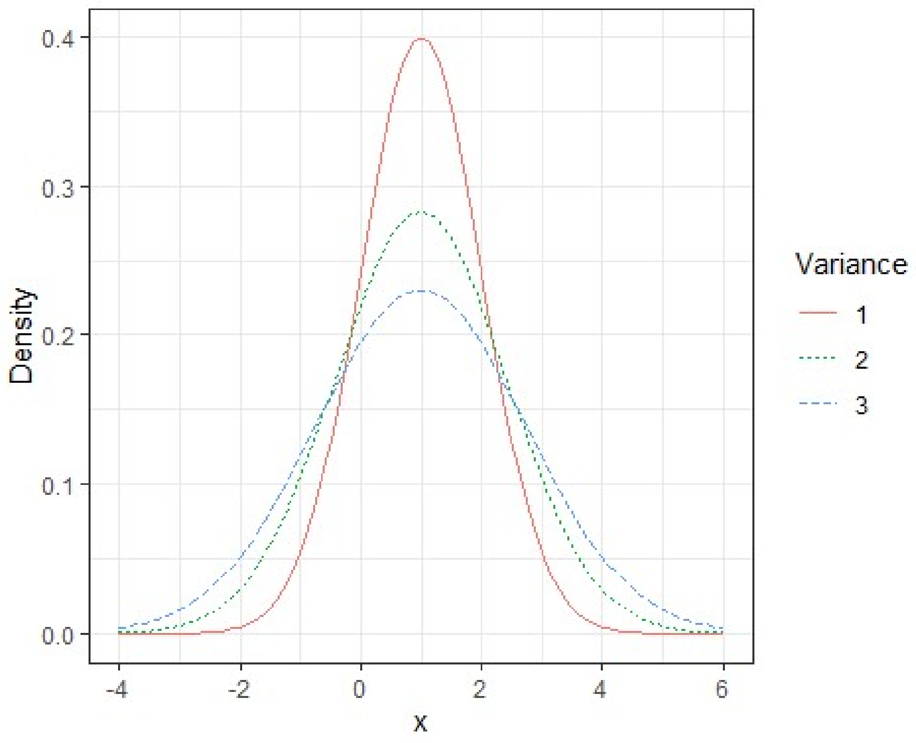
Also, the probability density function of a Gaussian random variable was introduced in Lecture 14.

(Q1) Generate a plot which displays the **probability density function** for three Gaussian random variables $X_1 \sim \mathcal{N}(\mu_1, \sigma_1^2)$, $X_2 \sim \mathcal{N}(\mu_2, \sigma_2^2)$, and $X_3 \sim \mathcal{N}(\mu_3, \sigma_3^2)$ with $\mu_1 = \mu_2 = \mu_3 = 1$ and $\sigma_1^2 = 1, \sigma_2^2 = 2, \sigma_3^2 = 3$.

Answer

```r
# the density function; Alternatively, you can use the function dnorm
f_musigma <- function(mu, sigma, x){
  y = (1/sigma/sqrt(2*pi)) * exp(-0.5*((x-mu)/sigma)^2)
  return (y)
}

df_density <- data.frame( x = seq(-4, 6, 0.1) ) %>%
  mutate( '1' = f_musigma(mu=1,sigma=1,x=x),
          '2' = f_musigma(mu=1,sigma=sqrt(2),x=x),
          '3' = f_musigma(mu=1,sigma=sqrt(3),x=x) )

df_density_longer = pivot_longer(df_density, col = c('1','2','3'),
                                 names_to = 'Variance', values_to = 'Density')
```

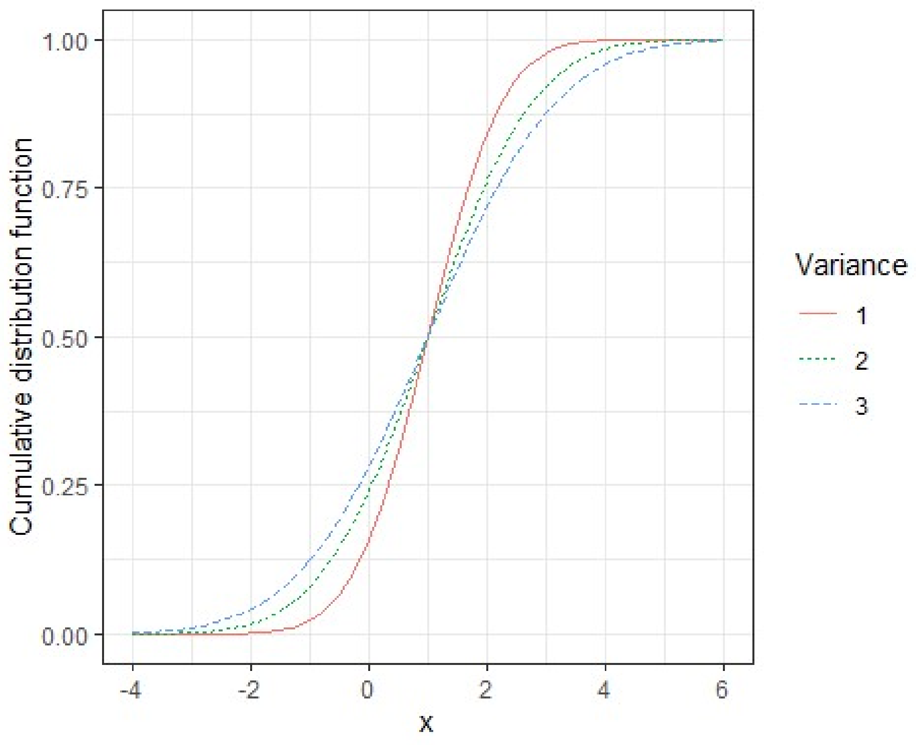
```r
ggplot(df_density_longer, aes(x=x, y = Density, linetype=Variance,
color=Variance)) +
  geom_line() + theme_bw()
```



(Q2) Generate a plot which displays the **cumulative distribution function** for three Gaussian random variables $X_1 \sim \mathcal{N}(\mu_1, \sigma_1^2)$, $X_2 \sim \mathcal{N}(\mu_2, \sigma_2^2)$, and $X_3 \sim \mathcal{N}(\mu_3, \sigma_3^2)$ with $\mu_1 = \mu_2 = \mu_3 = 1$ and $\sigma_1^2 = 1, \sigma_2^2 = 2, \sigma_3^2 = 3$.

Answer

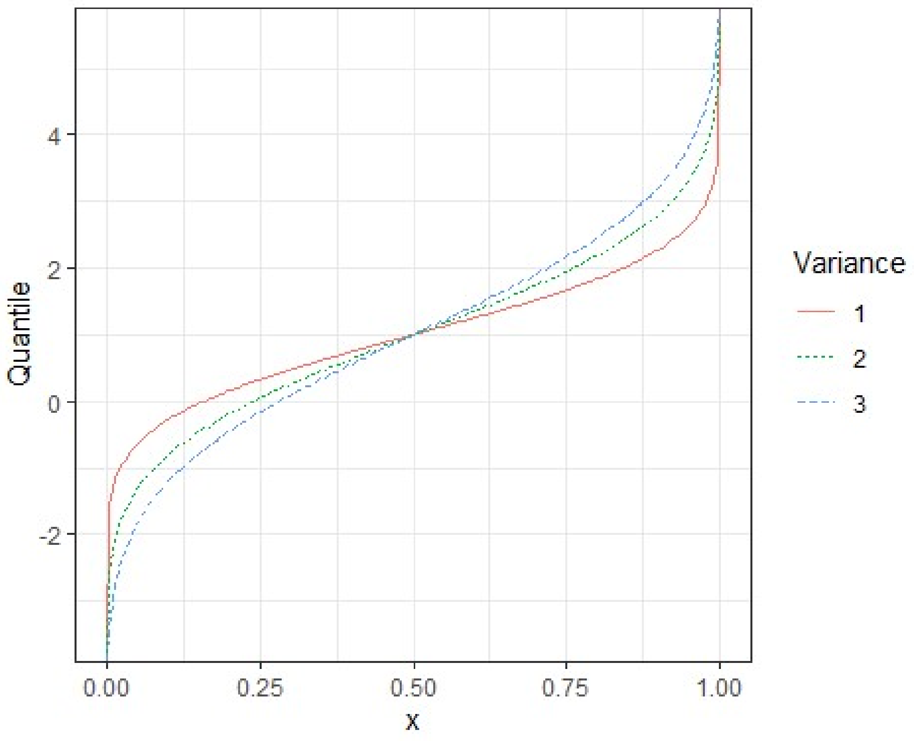
```r
df <- data.frame( x = seq(-4, 6, 0.1) ) %>%
  mutate( '1' = pnorm(mean=1,sd=1,q=x),
          '2' = pnorm(mean=1,sd=sqrt(2),q=x),
          '3' = pnorm(mean=1,sd=sqrt(3),q=x) )

df_longer = pivot_longer(df, col = c('1','2','3'),
                          names_to = 'Variance', values_to = 'CDF')

ggplot(df_longer, aes(x=x, y = CDF, linetype=Variance, color=Variance))
+
  geom_line() + theme_bw() + ylab('Cumulative distribution function')
```



(Q3) Generate a plot for the **quantile function** for the same three Gaussian distributions as above. Describe the relationship between the quantile function and the cumulative distribution function.

Answer

```
df <- data.frame( x = seq(0, 1, 0.005) ) %>%
  mutate( '1' = qnorm(mean=1,sd=1,p=x),
          '2' = qnorm(mean=1,sd=sqrt(2),p=x),
          '3' = qnorm(mean=1,sd=sqrt(3),p=x) )

df_longer = pivot_longer(df, col = c('1','2','3'),
                         names_to = 'Variance', values_to = 'Quantile')

ggplot(df_longer, aes(x=x, y = Quantile, linetype=Variance,
color=Variance)) +
  geom_line() + theme_bw()
```



Answer The quantile function is the inverse of the cumulative distribution function in this case.

(Q4)

Now use “rnorm()” to generate a random independent and identically distributed sequence $Z_1, \cdots, Z_n \sim \mathcal{N}(0,1)$ so that each $Z_i \sim \mathcal{N}(0,1)$ has standard Gaussian distribution. Set $n = 100$. Make sure your code is reproducible by using the “set.seed()” function. Store your random sample in a vector called ““standardGaussianSample””.

Answer

```
set.seed(0)
standardGaussianSample = rnorm(100, mean=0, sd=1)
head(standardGaussianSample)

## [1]  1.2629543 -0.3262334  1.3297993  1.2724293  0.4146414 -
1.5399500
```

(Q5)

Suppose $Z \sim \mathcal{N}(0,1)$ is a Gaussian random variable. Take $\alpha, \beta \in \mathbb{R}$ and let $W: \Omega \to \mathbb{R}$ be the random variable given by $W = \alpha Z + \beta$. Then W is also a Gaussian random variable. We will use this fact to create samples of Gaussian random variables from samples of standard Gaussian random variables.

Use your existing sample stored in “standardGaussianSample” to generate a new sample of the form $Y_1, \cdots, Y_n \sim \mathcal{N}(1,3)$ with expectation $\mu = 1$ and population variance $\sigma^2 = 3$. The i-th observation in this sample should be of the form $Y_i = \alpha \cdot Z_i + \beta$, for appropriately chosen $\alpha, \beta \in \mathbb{R}$, where Z_i is the i-th observation in the sample “standardGaussianSample”. Store the generated sample of $Y_1, \cdots, Y_n$ in a vector called “mean1Var3GaussianSampleA”. So to answer this question you need to decide the value of α and β, such that Y_i has the required expectation and variance.

Answer

```
mean1Var3GaussianSampleA = standardGaussianSample*sqrt(3) + 1
head(mean1Var3GaussianSampleA)

## [1]  3.1875010  0.4349472  3.3032799  3.2039122  1.7181800 -
1.6672717
```

(Q6)

Reset the random seed to the same value as the one you used in (Q4) using the “set.seed()” function and generate an i.i.d. sample of the form $Y_1, \cdots, Y_n \sim \mathcal{N}(1,3)$ using the “rnorm()” function (instead of using “standardGaussianSample”). Store this sample in a vector called “mean1Var3GaussianSampleB”. Are the entries of the vectors “mean1Var3GaussianSampleA” and “mean1Var3GaussianSampleB” the same?

Answer

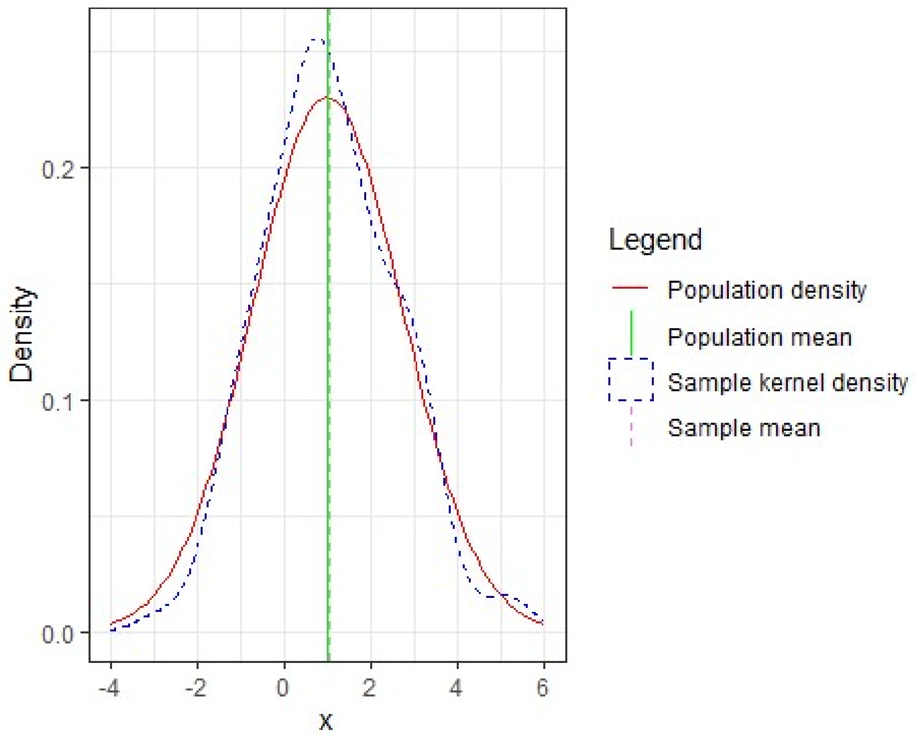
```
set.seed(0)
mean1Var3GaussianSampleB = rnorm(100, mean=1, sd=sqrt(3))
head(mean1Var3GaussianSampleB)

## [1]  3.1875010  0.4349472  3.3032799  3.2039122  1.7181800 -
1.6672717

all.equal(mean1Var3GaussianSampleA, mean1Var3GaussianSampleB)

## [1] TRUE
```

(Q7)

Now generate a graph which includes both a for your sample “mean1Var3GaussianSampleA” and a plot of the population density (the probability density function) generated using “dnorm()”. You can also include two vertical lines which display respectively the population mean and the sample mean.

Some guidance for creating the plot: It would be helpful to look at the example provided in Section 3.3(Q4) of Assignment 5. You may want to use the "geom_density()" and "geom_vline()" functions. In particular, both "geom_density()" and "geom_line()" have an argument called "data" that you may want to explore. Also, you can specify your own color by using the "scale_color_manual" and your own line type by "scale_linetype_manual".

```
colors<-c("Population density"="red", "Sample kernel density"="blue",
          "Population mean"="green", "Sample mean"="violet")
linetypes<-c("Population density"="solid", "Sample kernel
density"="dashed",
             "Population mean"="solid", "Sample mean"="dashed")

ggplot() + labs(x="x",y="Density") + theme_bw() +
  # 1. create plot of theoretical density
  geom_line(data=select(df_density, x, "3"),
            aes(x=x,y=`3`,color="Population density",
                linetype="Population density")) +
  # 2. add in kernel density plot from real sample
  geom_density(data=data.frame(x=mean1Var3GaussianSampleA),
               aes(x=x,color="Sample kernel density",
                   linetype="Sample kernel density")) +
  # 3. vertical lines
  geom_vline(aes(xintercept=1,color="Population mean",
                 linetype="Population mean")) +
  geom_vline(aes(xintercept=mean(mean1Var3GaussianSampleA),
                 color="Sample mean",linetype="Sample mean")) +
  # 4.
  scale_color_manual(name = "Legend", values=colors) +
  scale_linetype_manual(name="Legend", values=linetypes)
```



~~**(Q8)** (*)~~

This is an optional question (*). If you are short on time you can work on the other questions first.

Recall that for a random variable $X: \Omega \to \mathbb{R}$ is said to be Gaussian with expectation μ and variance σ^2 (i.e., $X \sim \mathcal{N}(\mu, \sigma^2)$) if for any $a, b \in \mathbb{R}$, we have

$$\mathbb{P}(a \leq X \leq b) = \int_a^b \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{z-\mu}{\sigma}\right)^2} dz.$$

Suppose $Z \sim \mathcal{N}(0,1)$ is a Gaussian random variable. Take $\alpha, \beta \in \mathbb{R}$ and let $W: \Omega \to \mathbb{R}$ be the random variable given by $W = \alpha Z + \beta$. In (Q5) we have assumed that W constructed in this way is a Gaussian random variable. Now, apply a change of variables to show that W is a Gaussian random variable with expectation β and variance α^2.

Hint: can you derive the expression of $\mathbb{P}(c \leq W \leq d)$?

Answer

$$\begin{aligned}\mathbb{P}(a \leq W \leq b) &= \mathbb{P}(a \leq \alpha Z + \beta \leq b) = \mathbb{P}\left(\frac{a-\beta}{\alpha} \leq Z \leq \frac{b-\beta}{\alpha}\right) \\ &= \int_{\frac{a-\beta}{\alpha}}^{\frac{b-\beta}{\alpha}} \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}z^2} dz\end{aligned}$$

Let $z = \frac{w-\beta}{\alpha}$, so $\frac{dz}{dw} = \frac{1}{\alpha}$. Apply a change of variables,

$$\begin{aligned}\mathbb{P}(a \leq W \leq b) &= \int_{\frac{a-\beta}{\alpha}}^{\frac{b-\beta}{\alpha}} \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}z^2} dz \\ &= \int_a^b \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{w-\beta}{\alpha}\right)^2} \cdot \left(\frac{dz}{dw}\right) dw \\ &= \int_a^b \frac{1}{\alpha\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{w-\beta}{\alpha}\right)^2} dw\end{aligned}$$

Therefore, W is a Gaussian random variable with expectation β and variance α^2.

2. Location estimators with Gaussian data

In this question we compare two estimators for the population mean μ_0 in a Gaussian setting in which we have independent and identically distributed data $X_1, \cdots, X_n \sim \mathcal{N}(\mu_0, \sigma_0^2)$.

The following code generates a data frame consisting of the mean squared error of the sample median as an estimator of μ_0.

```r
set.seed(0)
num_trials_per_sample_size <- 1000
min_sample_size <- 30
max_sample_size <- 500
sample_size_inc <- 5
mu_0 <- 1
sigma_0 <- 3

# create data frame of all pairs of sample_size and trial
simulation_df<-crossing(trial=seq(num_trials_per_sample_size),
                        sample_size=seq(min_sample_size,

max_sample_size,sample_size_inc)) %>%
  # simulate sequences of Gaussian random variables
  mutate(simulation=pmap(.l=list(trial,sample_size),
                         .f=~rnorm(.y,mean=mu_0,sd=sigma_0))) %>%
  # compute the sample medians
  mutate(sample_md=map_dbl(.x=simulation,.f=median)) %>%
  group_by(sample_size) %>%
  summarise(msq_error_md=mean((sample_md-mu_0)^2))
```

(Q1) Derive the mathematical expression for the population median of a Gaussian random variable $X_i \sim \mathcal{N}(\mu_0, \sigma_0^2)$.

Answer

Since

$$\begin{aligned}\mathbb{P}(X_i \le \mu_0) &= \int_{-\infty}^{\mu_0} \frac{1}{\sigma_0\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{z-\mu_0}{\sigma_0}\right)^2} dz \\ &= \int_{\mu_0}^{\infty} \frac{1}{\sigma_0\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{z-\mu_0}{\sigma_0}\right)^2} dz \\ &= \mathbb{P}(X_i \ge \mu_0),\end{aligned}$$

we have $\mathbb{P}(X_i \le \mu_0) = \mathbb{P}(X_i \ge \mu_0) = 0.5$, and therefore the cumulative function $F_X(\mu_0) = 0.5$. The population median is equal to the 0.5-quantile and hence is equal to μ_0.

(Q2)

Modify the above code to include estimates of the mean square error of the sample mean. Your data frame "simulation_df" should have a new column called "msq_error_mn" which estimates the mean squared error of the sample mean as an estimator of μ_0.

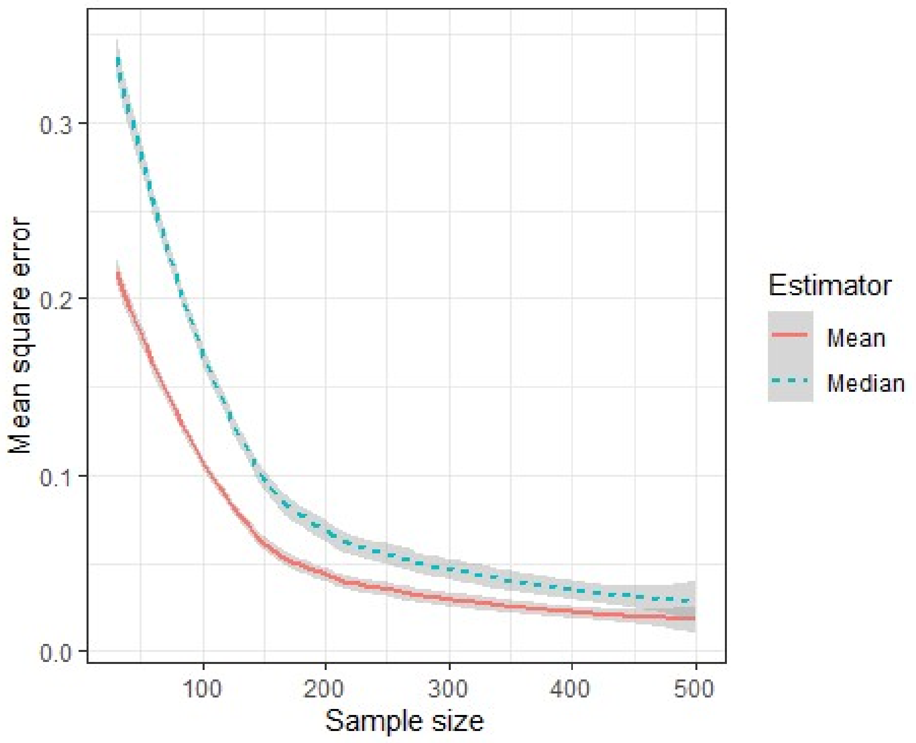
Then generate a plot which includes both the mean square error of the sample mean and the sample median as a function of the sample size.

–>

Answers:

```
set.seed(0)
num_trials_per_sample_size <- 1000
min_sample_size <- 30
max_sample_size <- 500
sample_size_inc <- 5
mu_0 <- 1
sigma_0 <- 3

# create data frame of all pairs of sample_size and trial
simulation_df<-crossing(trial=seq(num_trials_per_sample_size),
                        sample_size=seq(min_sample_size,

max_sample_size,sample_size_inc)) %>%
  # simulate sequences of Gaussian random variables
```

```r
  mutate(simulation=pmap(.l=list(trial,sample_size),
                         .f=~rnorm(.y,mean=mu_0,sd=sigma_0))) %>%
  # compute the sample medians
  mutate(sample_md=map_dbl(.x=simulation,.f=median)) %>%
  mutate(sample_mn=map_dbl(.x=simulation,.f=mean)) %>%
  group_by(sample_size) %>%
  summarise(msq_error_md=mean((sample_md-mu_0)^2),
msq_error_mn=mean((sample_mn-mu_0)^2))

simulation_df %>%
  pivot_longer(cols=c(msq_error_md,msq_error_mn),
               names_to="Estimator",values_to="msq_error") %>%
  mutate(Estimator=case_when(Estimator=="msq_error_md"~"Median",
                             Estimator=="msq_error_mn"~"Mean")) %>%

ggplot(aes(x=sample_size,y=msq_error,color=Estimator,linetype=Estimator
)) +
  geom_smooth()+theme_bw()+xlab("Sample size")+ylab("Mean square
error")
```



```r
# in this code, the name msq_error_md was changed by using the
case_when()
# alternatively, you can also use rename() before pivot_longer to do
that
```

~~3. (**) The law of large numbers and Hoeffding's inequality~~

This is an optional question. You can answer the other questions first before working on this one.

(Q1) Prove the following version of the weak law of large numbers.

Theorem (A law of large numbers). Let $X: \Omega \to \mathbb{R}$ be a random variable with a well-behaved expectation $\mu := \mathbb{E}(X)$ and variance $\sigma^2 := \mathrm{Var}(X)$. Let $X_1, \cdots, X_n: \Omega \to \mathbb{R}$ be a sequence of independent copies of X. Then for all $\epsilon > 0$,

$$\lim_{n\to\infty} \mathbb{P}\left(\left|\frac{1}{n}\sum_{i=1}^{n} X_i - \mu\right| \geq \epsilon\right) = 0.$$

You may want to begin by looking up the Chebyshev's inequality: For any random variable Z with finite expectation $\mathbb{E}(Z)$ and variance $\mathrm{Var}(Z)$, we have $\mathbb{P}(|Z - \mathbb{E}(Z)| \geq t) \leq \mathrm{Var}(Z)/t^2$ for any given number $t > 0$.

(Comparing the weak law of large numbers with Hoeffding's inequality): Below is some further information about Hoeffding's inequality. Hoeffding's inequality is the following important result:

Theorem (Hoeffding). Let $X: \Omega \to [0,1]$ be a **bounded** random variable with a well-behaved expectation $\mu := \mathbb{E}(X)$. Let $X_1, \cdots, X_n: \Omega \to \mathbb{R}$ be a sequence of independent copies of X. Then for all $\epsilon > 0$,

$$\mathbb{P}\left(\left|\frac{1}{n}\sum_{i=1}^{n} X_i - \mu\right| \geq \epsilon\right) \leq e^{-2n\epsilon^2}.$$

Please note that in the Hoeffding Theorem we additionally assume X to be bounded.

We can view Hoeffding's inequality as a variant of the law of large numbers. However, Hoeffding's inequality gives us information about the rate of convergence. In particular, the sample average for bounded random variables converges **exponentially** fast to its expectation.

Hoeffding's inequality is a precursor to Vapnik-Chervonekis theory which serves as a foundation for the theory of Statistical Machine Learning.

Answer

Since each X_i is an independent copy of X we have $\mathbb{E}(X_i) = \mu$ and $\mathrm{Var}(X_i) = \sigma^2$. We shall apply Chebyshev's inequality with $Z = \frac{1}{n}\sum_{i=1}^{n} X_i$. Note that by the linearity of expectation we have

$$\mathbb{E}(Z) = \mathbb{E}\left(\frac{1}{n}X_i\right) = \frac{1}{n}\sum_{i=1}^{n}\mathbb{E}\left(X_i\right) = \mu.$$

By the independent property of $X_1, X_2, \cdots, X_n$, we have

$$\mathrm{Var}(Z) = \mathrm{Var}\left(\frac{1}{n}X_i\right) = (1/n)^2 \cdot \sum_{i=1}^{n}\mathrm{Var}\left(X_i\right) = \sigma^2/n.$$

Hence, by applying Chebyshev's inequality with $t = \epsilon$ we have

$$\mathbb{P}\left(\left|\frac{1}{n}\sum_{i=1}^{n}X_i - \mu\right| \geq \epsilon\right) = \mathbb{P}(|Z - \mathbb{E}(Z)| \geq \epsilon) \leq \frac{\mathrm{Var}(Z)}{\epsilon^2} = \frac{\sigma^2}{n \cdot \epsilon^2}.$$

Letting $n \to \infty$ gives the required result.

4. Maximum likelihood estimates

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

4.1 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 extract a subset of the Hawks data set so that every Hawk in the subset 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 $\hat{\sigma}_{\text{MLE}}^2 = \frac{1}{n}\sum_{i=1}^{n}(X_i - \hat{\mu}_{\text{MLE}})^2$.

Apply the maximum likelihood method to compute the estimates $\hat{\mu}_{\text{MLE}}$ and $\hat{\sigma}_{\text{MLE}}^2$ 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;
```

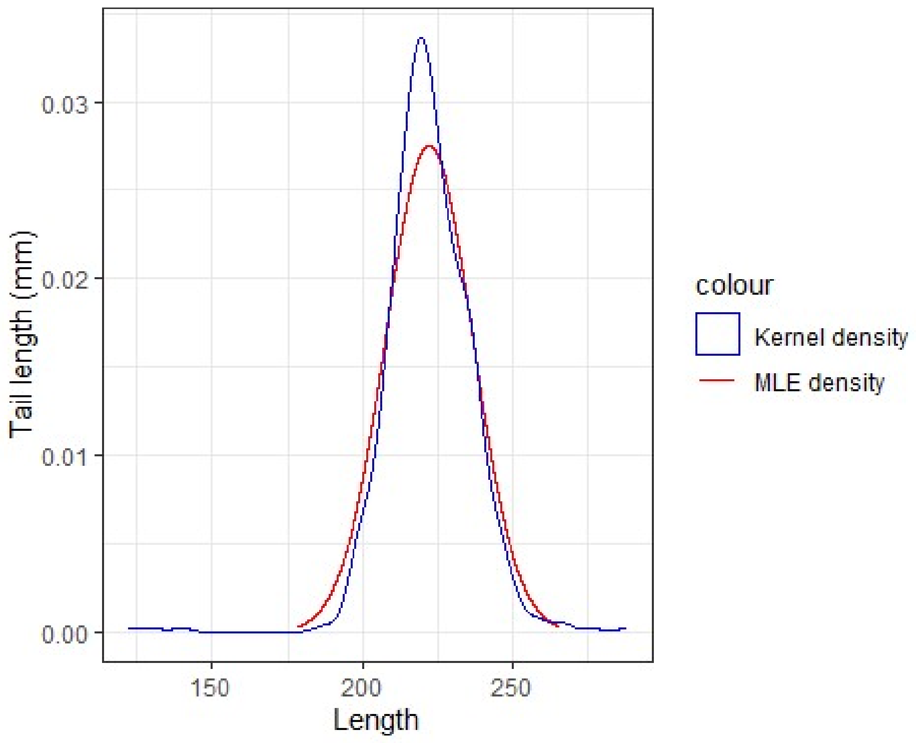
(Q3)

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))
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)
```



4.2 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 $\hat{V}_{\text{MLE}} = \frac{1}{n}\sum_{i=1}^{n}\left(X_i - \overline{X}\right)^2$ and let $\hat{V}_U := \frac{1}{n-1}\sum_{i=1}^{n}\left(X_i - \overline{X}\right)^2$ (recall that this is an unbiased estimator for variance (see Lecture 15), which is different from the MLE $\hat{\sigma}_{\text{MLE}}^2$ discussed in the last question).

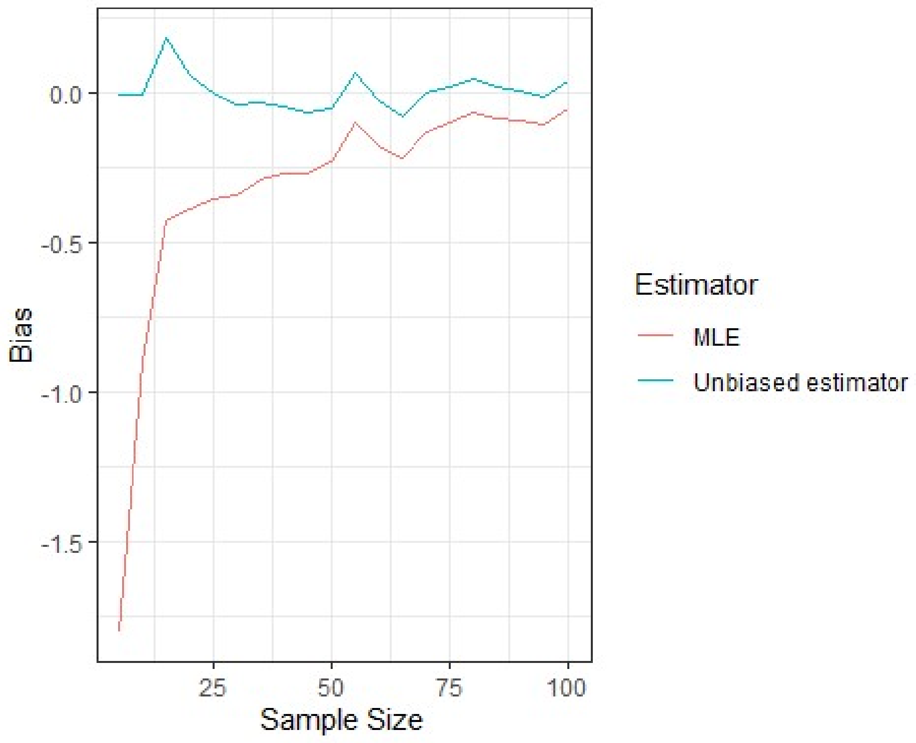
(Q1)

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 computed 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)

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')
```

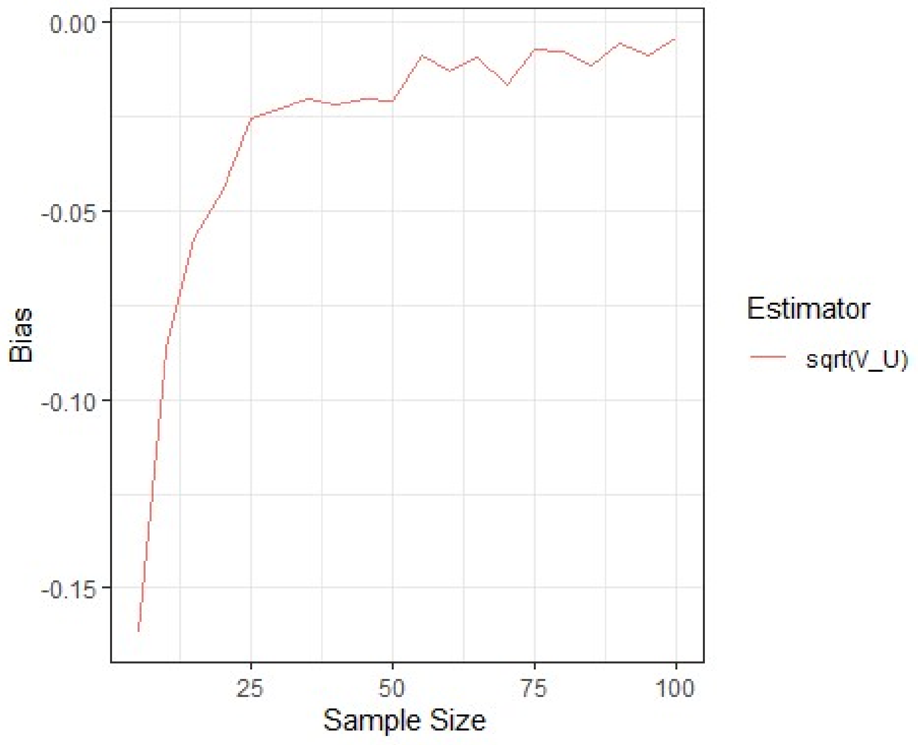


(Q2)

Is $\sqrt{\widehat{V}_U} = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}\left(X_i - \overline{X}\right)^2}$ 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
df_bias <- df %>% group_by(sample_size) %>%
  summarise(V_U_sqrt_bias=mean(V_U_sqrt)-sigma_0)

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 $\mathbb{E}\left(\sqrt{\hat{V}_U}\right)$ 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

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

By Jensen's inequality

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

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 the bias of $\hat{V}_U$ (not using R programming.)~~

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

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

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

4.3 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

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

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

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

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 $\hat{\lambda}_{\mathrm{MLE}} = \overline{X}$.

Answer

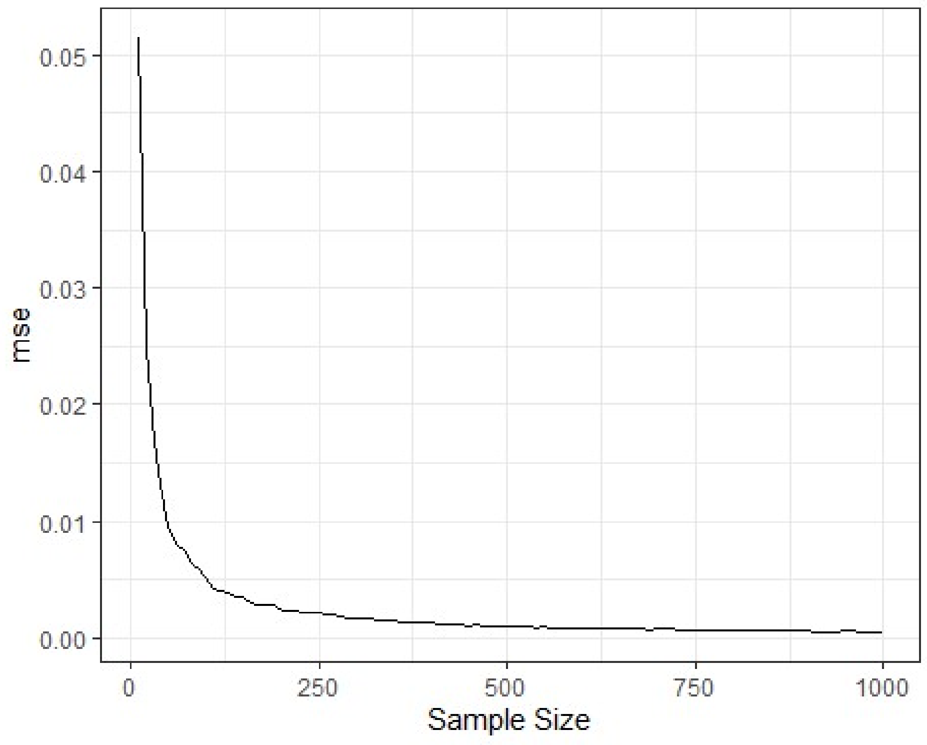
The derivative $\frac{\partial}{\partial\lambda}\log l(\lambda)$ is bigger than zero when $\lambda < \overline{X}$, and less than zero when $\lambda > \overline{X}$. So $\lambda \to \log l(\lambda)$ reaches its maximum at the single point at which $\lambda = \overline{X}$. Hence, the maximum likelihood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

(Q3)

Now conduct a simulation experiment which explores the behaviour of $\hat{\lambda}_{\mathrm{MLE}}$ 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

```
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 ))

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 $\hat{\lambda}_{\mathrm{MLE}}$ 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

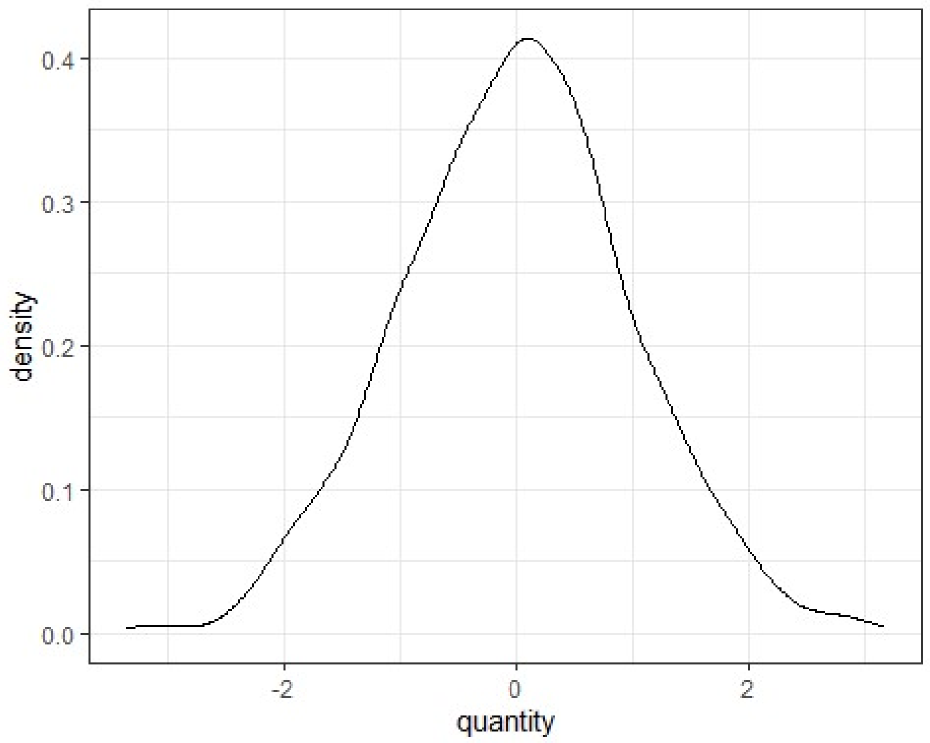
$$\frac{\partial}{\partial\lambda}\log p_\lambda(X) = -n + \frac{X}{\lambda}$$

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

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

```r
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.

4.4 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 $p_\lambda\colon \mathbb{R} \to (0, \infty)$ define by

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

(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 $\hat{\lambda}_{\mathrm{MLE}} = 1/\overline{X}$.

(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

```
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 $\hat{\lambda}_{\mathrm{MLE}}$ from your data.

Answer

```
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

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

## [1] 0.2997893
```

5. Confidence intervals

5.1 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.

```
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()

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)

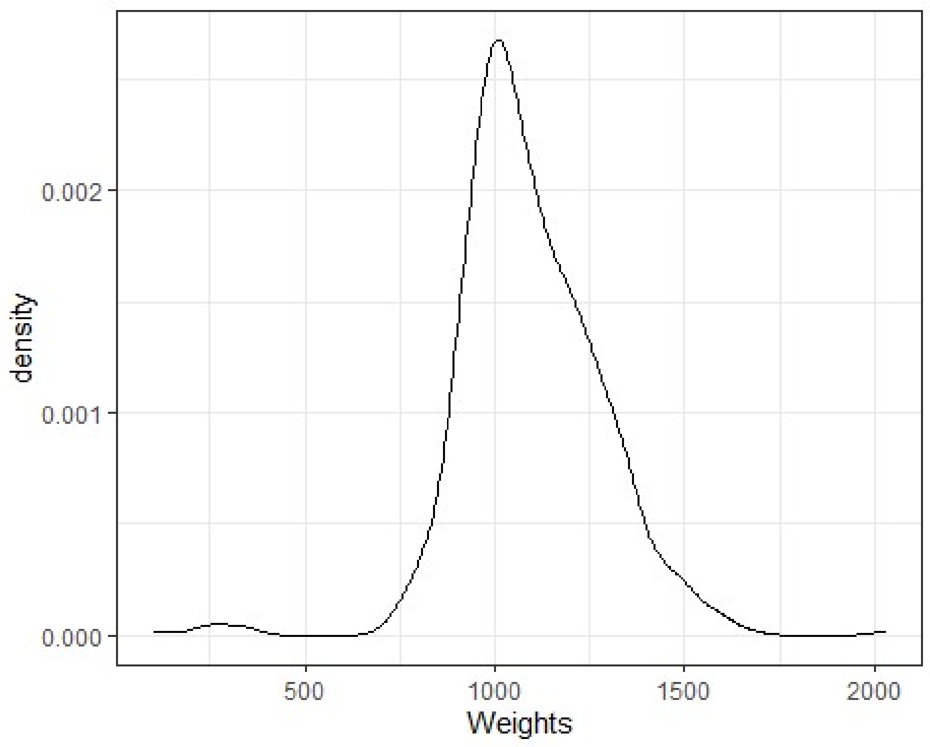
What assumptions are made to derive confidence intervals based on Student's t-distribution? Check if these assumptions are justified using a kernel density plot with the "geom_density()" function and using a "QQ"-plot with the "stat_qq"() function.

Answer

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

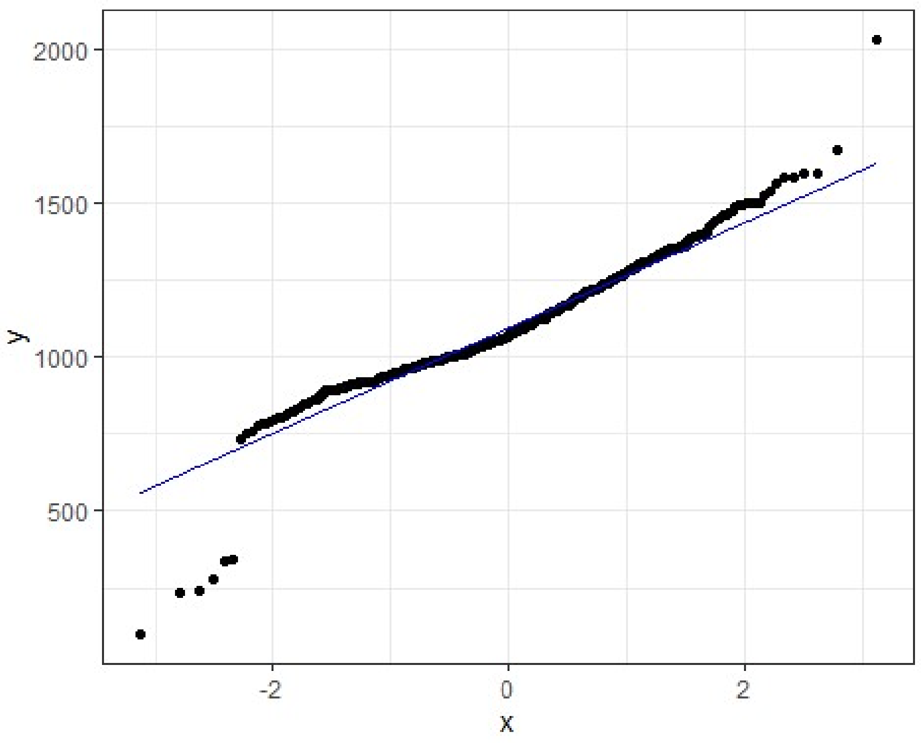
First we create a kernel density plot. This reveals that the distribution is unimodal but has slightly heavier tails than a Gaussian.

```r
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.

```r
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.

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

## [1] 577
```

5.2 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 $S := \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}\left(X_i - \overline{X}\right)^2}$ be the sample standard deviation. In addition, let $t_{\alpha/2,n-1}$ be the $\left(1-\frac{\alpha}{2}\right)$-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 $\left(L_\alpha(X_1, \cdots, X_n), R_\alpha(X_1, \cdots, X_n)\right)$ defined by

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

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 $\left(L_\alpha(X_1,\cdots,X_n), R_\alpha(X_1,\cdots,X_n)\right)$

```
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:

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

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. $\gamma = 0.95$.

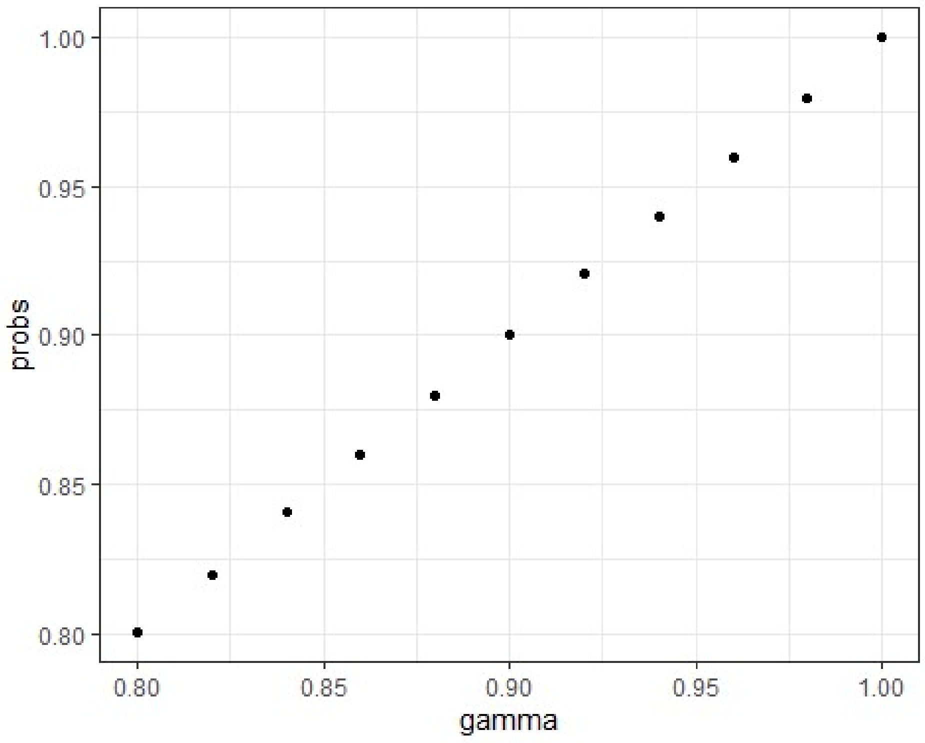
```
num_trials <- 100000
sample_size <- 30
mu_0 <- 1
sigma_0 <- 3
alpha <- 0.05
set.seed(0) # set random seed for reproducibility

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

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

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))

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



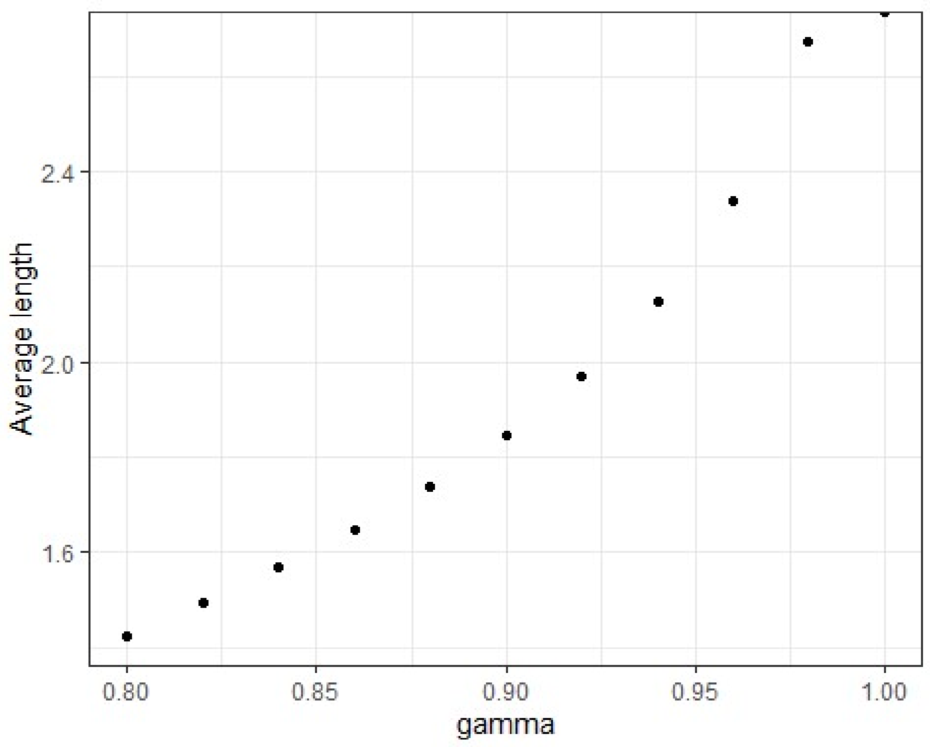
The results suggest that $\mathbb{P}\{(L_\alpha(X_1,\cdots,X_n) \le \mu_0 \le R_\alpha(X_1,\cdots,X_n))\}$ is very close to γ. In fact $\mathbb{P}\{(L_\alpha(X_1,\cdots,X_n) \le \mu_0 \le R_\alpha(X_1,\cdots,X_n))\} = \gamma$.

(Q2)

How does the average length $\mathbb{E}(R_\alpha(X_1,\cdots,X_n) - L_\alpha(X_1,\cdots,X_n))$ 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:
  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))

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.