Assignment 6

EMATM0061: Statistical Computing and Empirical Methods, TB1, 2022

Dr. Rihuan Ke

Introduction

This is the sixth 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 14 and 15 (see the Blackboard).

The submission deadline for this assignment is 23:59, 14 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 `Assignment06_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/YqhAnj5sPY

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. Continuous random variables and limit laws

This section is mainly based on Lecture 14. We will explore several continuous random variables and the limiting behaviors of sequences of i.i.d. random variables.

1.1 Simulating data with the uniform distribution

Suppose that $\alpha, \beta \in [0,1]$ with $\alpha + \beta \leq 1$ and let X be a discrete random variable with distribution supported on $\{0, 3, 10\}$. Suppose that $\mathbb{P}(X = 3) = \alpha$ and $\mathbb{P}(X = 10) = \beta$, $\mathbb{P}(X = 0) = 1 - \alpha - \beta$, and $\mathbb{P}(X \notin \{0, 3, 10\}) = 0$.

We will use the uniform distribution to simulate data for the discrete random variable X. A standard uniformly distributed random variable U is a continuous random variable with probability density function

$$p_U(x) = \begin{cases} 1 \text{ if } x \in [0,1], \\ 0 \text{ otherwise.} \end{cases}$$

(Q1). Show that for any pair of numbers $a, b \in \mathbb{R}$ with $0 \leq a \leq b \leq 1$, we have $\mathbb{P}(U \in [a,b]) = b - a$.

(Q2).

Now, let's consider the discrete random variable X (as defined above) and the case with $\alpha = \beta = 0.25$. You can generate a sequence of i.i.d. copies $X_1, X_2, \cdots, X_n$ of X as follows:

```
set.seed(0)
n <- 1000
sample_X <- data.frame(U=runif(n)) %>%
  mutate(X=case_when(
    (0<=U)&(U<0.25)~3,
    (0.25<=U)&(U<0.5)~10,
    (0.5<=U)&(U<=1)~0)) %>%
  pull(X)
```

Why does this `sample_X` correspond to a sequence of i.i.d. copies $X_1, X_2, \cdots, X_n$ of X where $\mathbb{P}(X = 3) = \alpha$ and $\mathbb{P}(X = 10) = \beta$, and $\mathbb{P}(X = 0) = 1 - \alpha - \beta$ with $\alpha = \beta = 0.25$? To answer this you may want to explain the distribution of the numbers/entries of `sample_X`, i.e., from which distribution they are drawn?

(Q3)

Now create a function called `sample_X_0310()` which takes as inputs `alpha`, `beta` and `n` and outputs a sample $X_1, X_2, \cdots, X_n$ of independent copies of X where $\mathbb{P}(X = 3) = \alpha$ and $\mathbb{P}(X = 10) = \beta$, and $\mathbb{P}(X = 0) = 1 - \alpha - \beta$ with $\alpha = \beta = 0.25$.

(Q4)

Next take $\alpha = 1/2$ and $\beta = 1/10$, and use your function `sample_X_0310()` to create a sample of size $n = 10000$ of the form $X_1, X_2, \cdots, X_n$ consisting of independent copies of X. What is the sample average of $X_1, X_2, \cdots, X_n$? How does this compare with the theoretical value of $\mathbb{E}(X)$ (We have worked on the expression for this expectation in assignment 5 Section 2.1)? Then use your understanding of the law of large numbers to explain this behavior.

(Q5)

Based on the sample generated from the last question, compute the sample variance of $X_1, X_2, \cdots, X_n$ and compare it with the population variance $\text{Var}(X)$ (again, the expression of $\text{Var}(X)$ has been derived in Assignment 5).

(Q6)

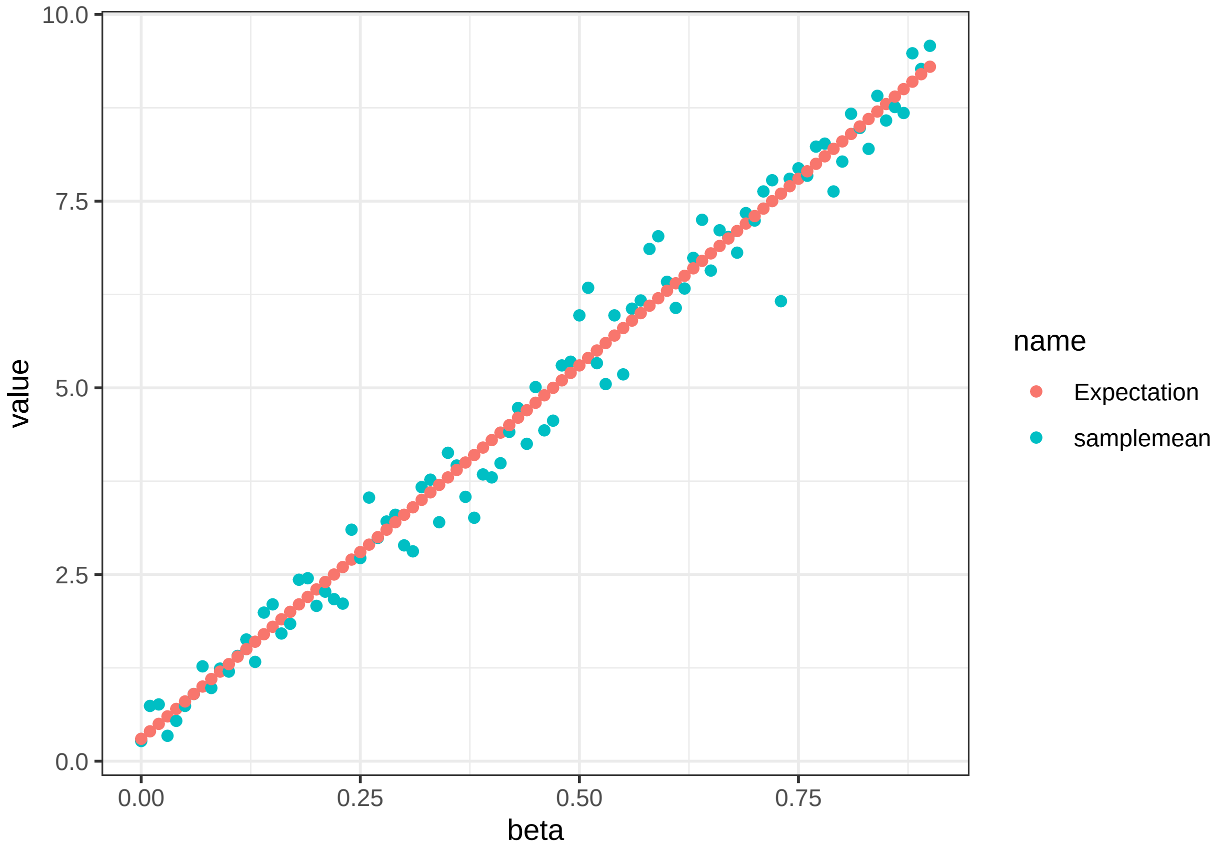
Now take $n = 100$, $\alpha = 1/10$ and vary β in increments of 0.01 from 0 to 9/10 (including 9/10). Create a data frame that has the following four columns. You can use the functions `mutate`, `map`, and `map_dbl` to help you.

1. The first column is called `beta` (contains different β values ranging from 0 to 9/10).
2. The second column is called `sample_X` including samples with the corresponding β. More specifically, for each value of β (in one of the rows), create a sample of $X_1, X_2, \cdots, X_n$ consisting of independent copies of X, by using your function `sample_X_0310()`.
3. The third column is called `samplemean`, which contains sample means of the samples.
4. The last column is called `Expectation`, which contains numerical values of the population mean $\mathbb{E}(X)$ (for the corresponding value of β in the same row).

(Q7)

Create a plot of the sample averages and $\mathbb{E}(X)$ as a function of β

Your plot should look similar to this:



1.2 Exponential distribution

Let $\lambda > 0$ be a positive real number. An exponential random variable X with rate parameter λ is a continuous random variable with density $p_\lambda : \mathbb{R} \to (0, \infty)$ defined by

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

(Q1)

Prove that p_λ is a well-defined probability density function. Then derive mathematical expressions for the cumulative distribution function and the quantile function for exponential random variables with parameter λ.

(Q2)

Now implement a function called `my_cdf_exp()`. The function `my_cdf_exp()` should take as input two numbers $x \in \mathbb{R}$ and $\lambda > 0$ and output the value of the cumulative distribution function $F_X(x)$ where X is an exponential random variable with rate parameter λ.

Check your function my_cdf_exp() gives rise to the following output:

```
lambda <- 1/2
map_dbl(.x=seq(-1,4), .f=~my_cdf_exp(x=.x,lambda=lambda) )
```

```
## [1] 0.0000000 0.0000000 0.3934693 0.6321206 0.7768698 0.8646647
```

Then type ?pexp into your R console to learn more about R's inbuilt cumulative distribution function for the exponential distribution. We can now confirm that our `my_cdf_exp` is correct when $\lambda = 1/2$ as follows:

```
test_inputs <- seq(-1,10,0.1)
my_cdf_output <- map_dbl(.x=test_inputs, .f=~my_cdf_exp(x=.x,lambda=lambda))
inbuilt_cdf_output <- map_dbl(.x=test_inputs,.f=~pexp(q=.x,rate=lambda))
all.equal(my_cdf_output,inbuilt_cdf_output)
```

```
## [1] TRUE
```

(Q3)

Next implement a function called `my_quantile_exp()`. The function `my_quantile_exp()` should take as input two arguments $p \in [0, 1]$ and $\lambda > 0$ and output the value of the quantile function $F_X^{-1}(p)$ where X is an exponential random variable with rate parameter λ.

Once you have implemented your function compare it with R's inbuilt qexp function using the same procedure as we used above for the cumulative distribution function for inputs $\lambda = 1/2$ and $p \in \{0.01, 0.02, \cdots, 0.99\}$. Note that you don't need to consider inputs $p \leq 0$ or $p \geq 1$ here.

(Q4)

From the probability density function, derive an expression for the population mean and variance of an exponential random variable X with parameter λ. You may want to use integration by parts when computing the integrals.

1.3 The Binomial distribution and the central limit theorem

Two important discrete distributions are the Bernoulli distribution and the Binomial distribution.

We say that a random variable X has Bernoulli distribution with parameter $p \in [0,1]$ if $\mathbb{P}(X=1)=p$ and $P(X=0)=1-p$. This is often abbreviated as $X \sim \mathcal{B}(p)$.

Given $n \in \{1,2,3,\cdots\}$ and $p \in [0,1]$, we say that a random variable Z has a Binomial distribution with parameters n and p if $Z = X_1 + \cdots + X_n$ for some random variables $\{X_i\}$ where $X_i \sim \mathcal{B}(p)$ and $X_1, \cdots, X_n$ are independent and identically distributed. This is often abbreviated as $Z \sim \text{Binom}(n,p)$.

(Q1)

Give an expression for the expectation and variance of $Z \sim \text{Binom}(n,p)$. You may want to make use of the following two useful facts:

1. Given any sequence of random variables $W_1, \cdots, W_k$ we have $\mathbb{E}(\sum_{i=1}^k W_i) = \sum_{i=1}^k \mathbb{E}(W_i)$.
2. Given **independent** random variables $W_1, \cdots, W_k$ we have $\text{Var}(\sum_{i=1}^k W_i) = \sum_{i=1}^k \text{Var}(W_i)$.

Note that the second fact may not hold if $W_1, \cdots, W_k$ are not independent.

(Q2)

The function `dbinom()` in R allows us to compute the probability mass function of a Binomial random variable $Z \sim \text{Binom}(n,p)$. For $x \in \{0,1,\cdots,n\}$, the function `dbinom(x,size=n,prob=p)` will return the value of the probability mass function evaluated at x, i.e., it returns $p_z(x) = \mathbb{P}(Z=x)$. You can run ?dbinom in the R console to find out more.

Consider the case where $n = 50$ and $p = 7/10$. Use the `dbinom()` to generate a data-frame called `binom_df` with two columns called `x` and `pmf`.

1. The first column contains the numbers $\{0,1,\cdots,50\}$ inclusive. These are different values of x.

2. The second column gives the corresponding value of the probability mass function $p_z(x) = \mathbb{P}(Z=x)$ with $Z \sim \text{Binom}(50, 0.7)$. Use the `head()` function to observe the first 3 rows of your data frame.

The result should look as follows:

```
##   x          pmf
## 1 0 7.178980e-27
## 2 1 8.375477e-25
## 3 2 4.787981e-23
```

(Q3)

The function `dnorm()` in R allows us to compute the probability density function of a Gaussian random variable $W \sim \mathcal{N}(\mu, \sigma^2)$ with expectation μ and variance σ^2. The function `dnorm(x,mean=mu,sd=sigma)` will return the probability density function evaluated at x, i.e., $f_W(x)$ for $W \sim \mathcal{N}(\mu,\sigma^2)$. You can run ?dnorm in the R console to find out more.

We shall consider a case where $\mu = 50 \cdot 0.7$ and $\sigma = \sqrt{50 \cdot 0.7 \cdot (1-0.7)}$. Use the `dnorm()` to generate a data-frame called `gaussian_df` with two columns called `x` and `pdf`.

1. The first column contains the numbers $0, 0.01, 0.02, \cdots, 50$. These numbers represent different values of x.
2. The second column gives the corresponding value of the probability density function $f_W(x)$ with $W \sim \mathcal{N}(\mu, \sigma^2)$. Use the `head()` function to observe the first 3 rows as your data frame.

Your result should look as follows:

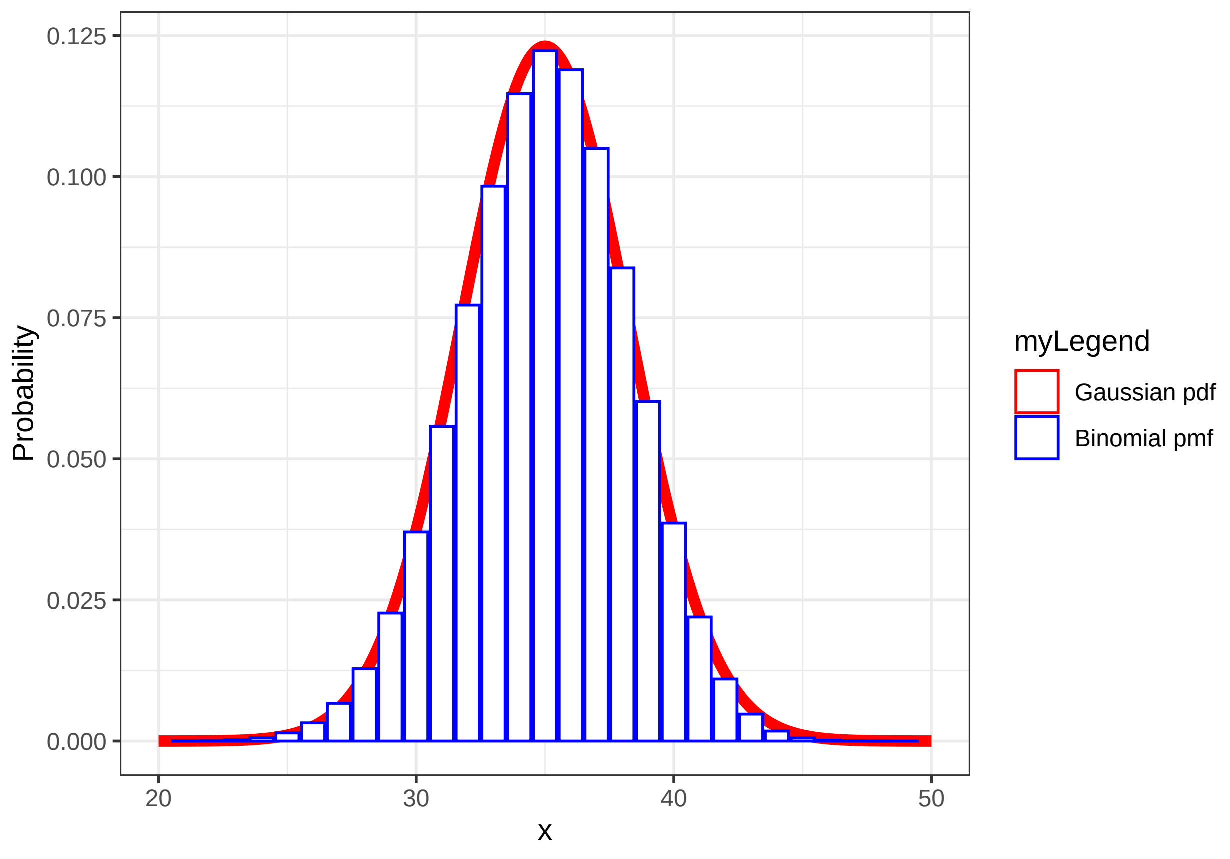
```
##      x          pdf
## 1 0.00 5.707825e-27
## 2 0.01 5.901264e-27
## 3 0.02 6.101201e-27
```

(Q4)

Next, based on the `binom_df` and `gaussian_df` you generated above, use the following code to create a plot which compares the probability density for your Gaussian distribution $W \sim \mathcal{N}(\mu, \sigma^2)$ where $\mu = n \cdot p$ and $\sigma = \sqrt{n \cdot p \cdot (1-p)}$ and the probability mass function for your Binomial distribution $Z \sim \text{Binom}(n, p)$. Try to use the central limit theorem to explain the results you observe.

```r
colors<-c("Gaussian pdf"="red", "Binomial pmf"="blue")
fill<-c("Gaussian pdf"="white", "Binomial pmf"="white")


ggplot() + labs(x="x",y="Probability") + theme_bw() +
  # create plot of Gaussian density
  geom_line(data=gaussian_df, aes(x,y=pdf,color="Gaussian pdf"),size=2) +
  # create a bar chart from PMF of Binomial distribution
  geom_col(data=binom_df, aes(x=x,y=pmf, color="Binomial pmf",fill="Binomial pmf")) +
  # set color
  scale_color_manual(name = "myLegend", values=colors) +
  scale_fill_manual(name = "myLegend", values=fill) +
  xlim(c(20,50))
```



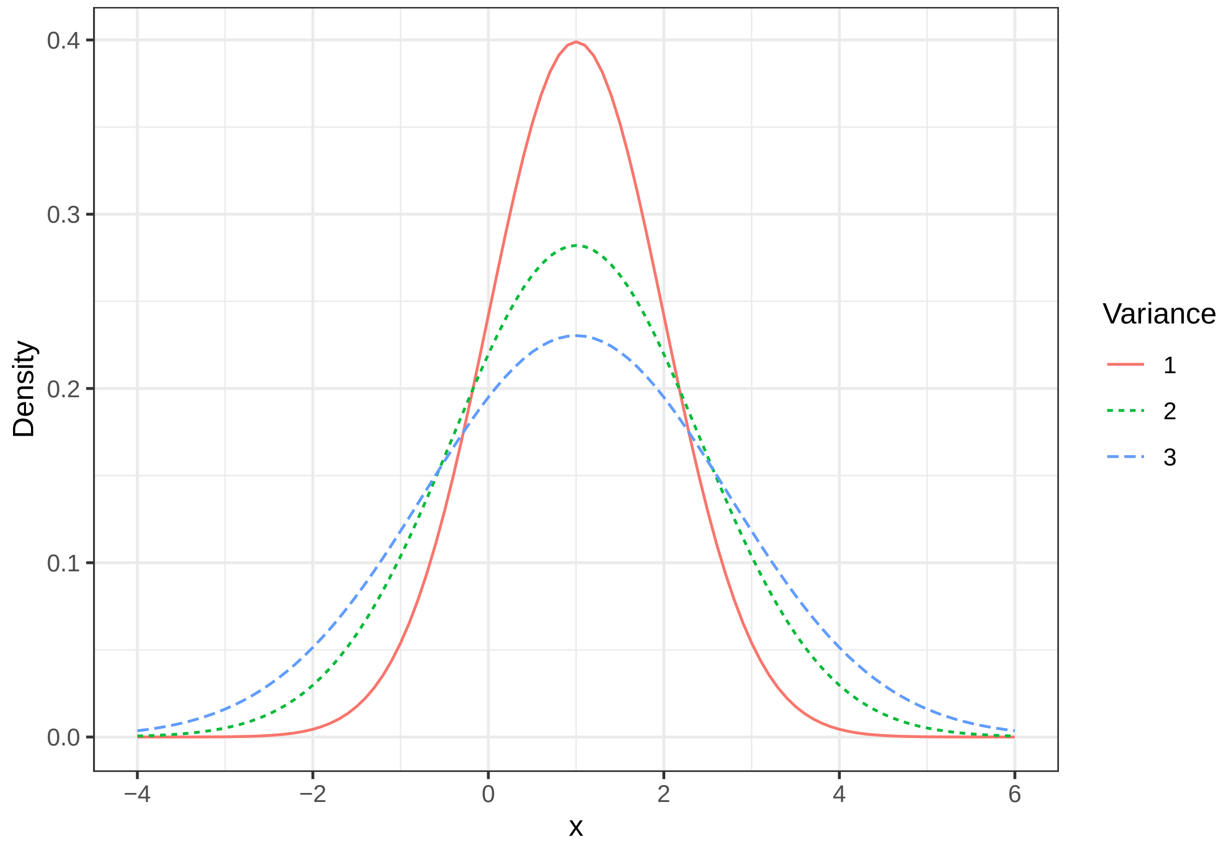
1.4 The Gaussian distribution

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 $\sigma_1^2 = 1, \sigma_2^2 = 2, \sigma_3^2 = 3$.

Your plot should look like this:



(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 $\sigma_1^2 = 1, \sigma_2^2 = 2, \sigma_3^2 = 3$.

(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.

(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`".

(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.

(Q6)

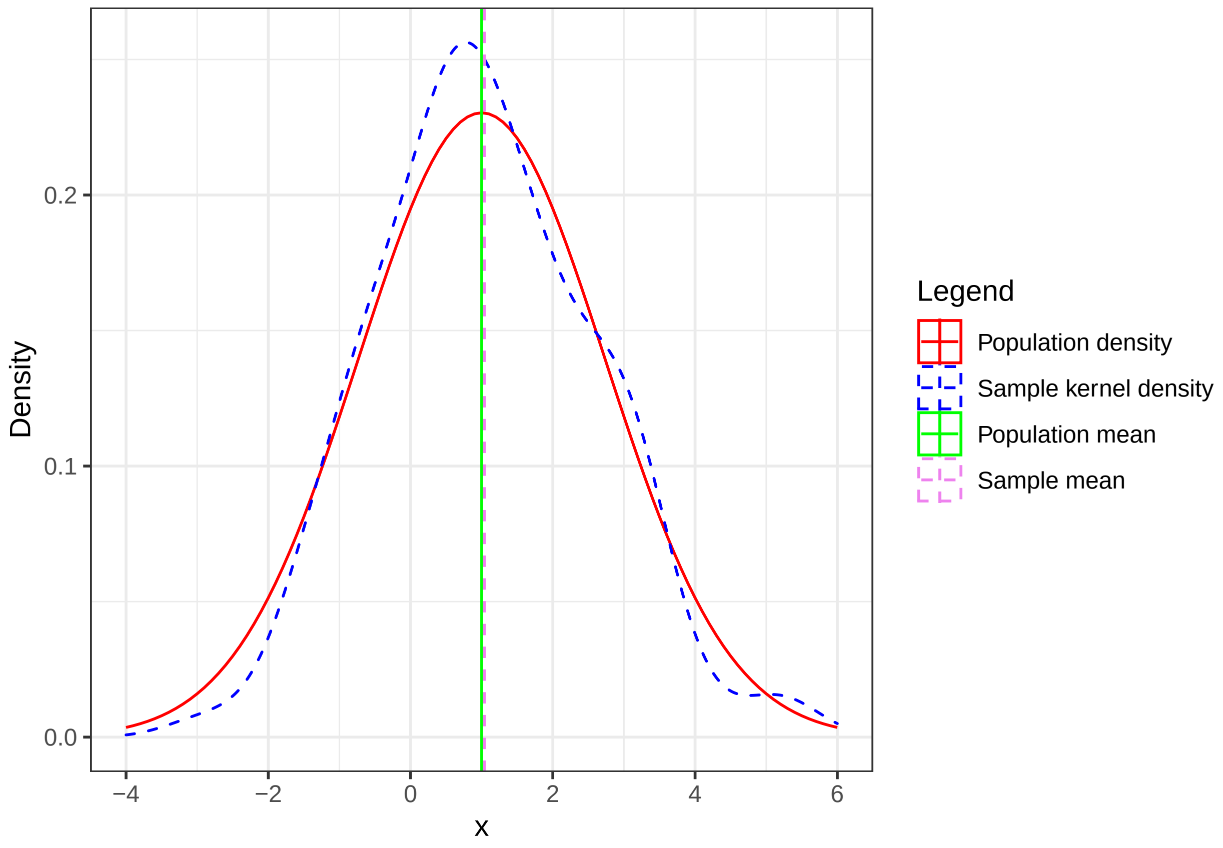
Reset the random seed to the same value as the one you used in (Q4) using the `set.seed()` function and generate an i.i.d. sample of the form $Y_1, \cdots, Y_n \sim \mathcal{N}(1, 3)$ using the `rnorm()` function. Store this sample in a vector called `mean1Var3GaussianSampleB`. Are the entries of the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

(Q7)

Now generate a graph which includes both a *kernel density plot* 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 1.3(Q4). You may want to use the `geom_density()` and `geom_vline()` functions. In particular, both `geom_density()` and `geom_vline()` 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`.

Your plot should look similar to the following:



(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

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

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.

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))
```

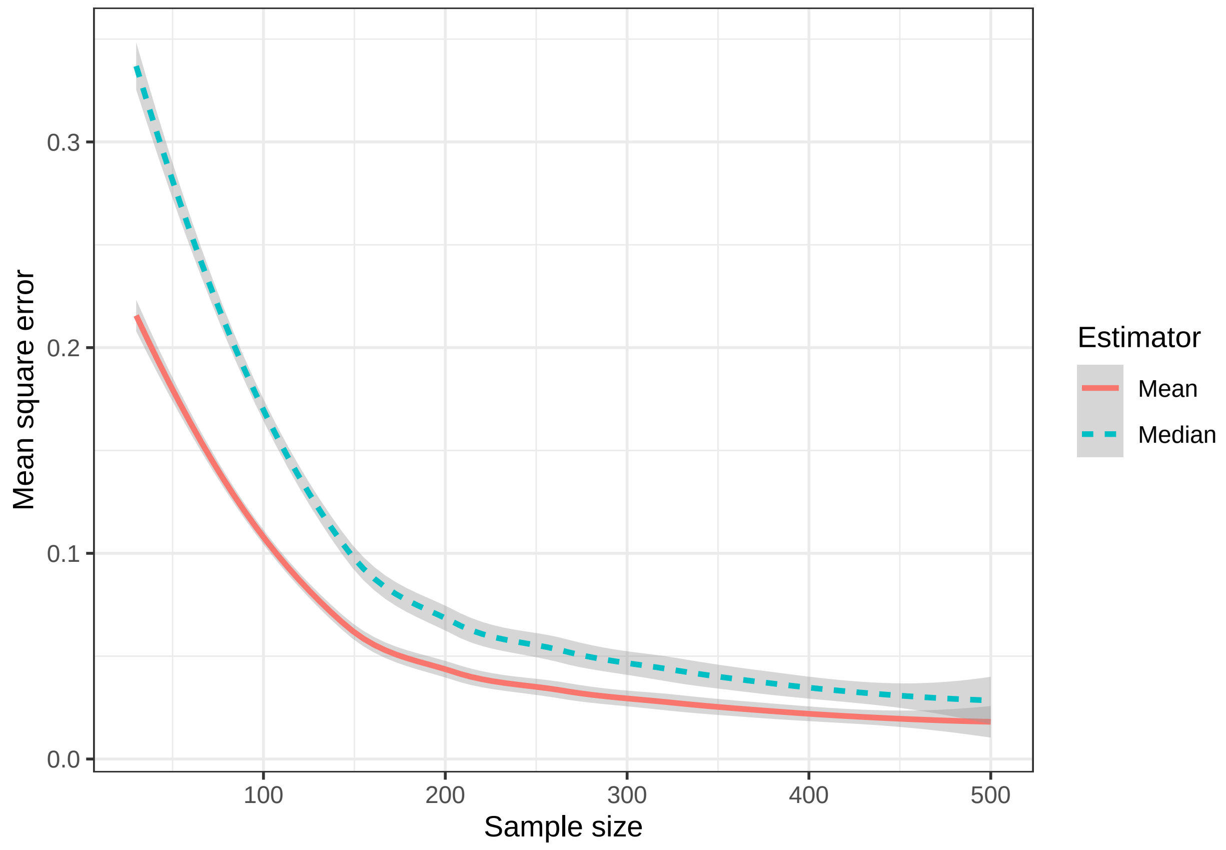
(Q1) What is the population median of a Gaussian random variable $X_i \sim \mathcal{N}(\mu_0, \sigma_0^2)$?

(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.

Your plot might look like the following:



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

This is an optional question.

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 $\epsilon > 0$,

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

You may want to begin by looking up the Chebyshev's inequality.

Below is some further information about Hoeffding's inequality, which is another version of the law of large numbers. Hoeffding's inequality is the following important result:

Theorem (Hoeffding). Let $X : \Omega \to \mathbb{R}$ be a 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 $\epsilon > 0$,

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

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.