Assignment 6

Vishal

2022-11-09

1. Continuous random variables and limit laws

1.1 (Q1)

We know that,
$\mathbb{P}(X \leq b) = \int_{-\infty}^{b} f_x(x)\,dx$
and
$\mathbb{P}(X \leq a) = \int_{-\infty}^{a} f_x(x)\,dx$
Which gives,
$\mathbb{P}(X \leq b) - \mathbb{P}(X \leq a) = \int_{a}^{b} f_x(x)\,dx$
i.e. $\mathbb{P}(X \leq b) - \mathbb{P}(X \leq a) = b - a$
Therefore,
$\mathbb{P}(U \in [a, b]) = b - a$

1.1 (Q2)

```
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)

head(sample_X)
```

```
## [1]  0 10 10  0  0  3
```

```
table(sample_X)
```

```
## sample_X
##   0   3  10
## 481 244 275
```

If we look at the $sample_x$ data frame, we see that, there are 481 samples that correspond to 0, 244 samples of 3 and 275 samples of 10. These are discretely random variables with uniform distribution.

These are generated randomly and so the number of occurrences may change every time this code is run.

To prove the distributions, we can integrate $\mathbb{P}_X(x)dx$ over the probabilities.
So,
$\mathbb{P}(X = 3) = \mathbb{P}_x(x)dx$
$\Rightarrow \int_0^{0.25} f_x(x)\,dx$
$\Rightarrow \int_0^{0.25} \alpha\,dx$
$\Rightarrow 0.25$

And
$\mathbb{P}(X = 10) = \mathbb{P}_x(x)dx$
$\Rightarrow \int_{0.25}^{0.50} f_x(x)\,dx$
$\Rightarrow \int_{0.25}^{0.50} \beta\,dx$
$\Rightarrow 0.25$

And
$\mathbb{P}(X = 0) = \mathbb{P}_x(x)dx$
$\Rightarrow \int_{0.5}^{1} f_x(x)\,dx$
$\Rightarrow \int_{0.5}^{1} x\,dx$
$\Rightarrow 0.5$ i.e. $1 - \alpha - \beta$

1.1 (Q3)

```
sample_x_0310 <- function(alpha, beta, n) {
  df <- data.frame(U = runif(n)) %>%
    mutate(X = case_when((0 <= U) & (U < alpha) ~ 3,
                         (alpha <= U) & (U < (alpha + beta)) ~ 10,
                         ((alpha + beta) <= U) & (U <= 1) ~ 0)) %>%
    pull(X)
  return(df)
}
```

1.1 (Q4)

```
df <- sample_x_0310(1/2, 1/10, 10000)
head(df)
```

```
## [1]  3 10  0  3  0  3
```

Sample average of $X_1, X_2, ..., X_n$:

```
mean(df)
```

```
## [1] 2.5135
```

Theoretical Expectation of $X_1, X_2, ..., X_n$:

```
vals <- c(3, 10, 0)
probs <- c(1/2, 1/10, 2/5)

Ex <- weighted.mean(vals, probs)
Ex
```

```
## [1] 2.5
```

Since the number of samples we have considered in the above example is 10000 which is very high, the theoretical expected value and the calculated expected values are similar. If we increase the number of samples, difference between the two will reduce further.

1.1 (Q5)

Variance of the sample:

```
var(df)
```

```
## [1] 8.328651
```

Theoretical variance:

```
Vx <- sum(((vals - Ex) ^ 2) * probs)
Vx
```

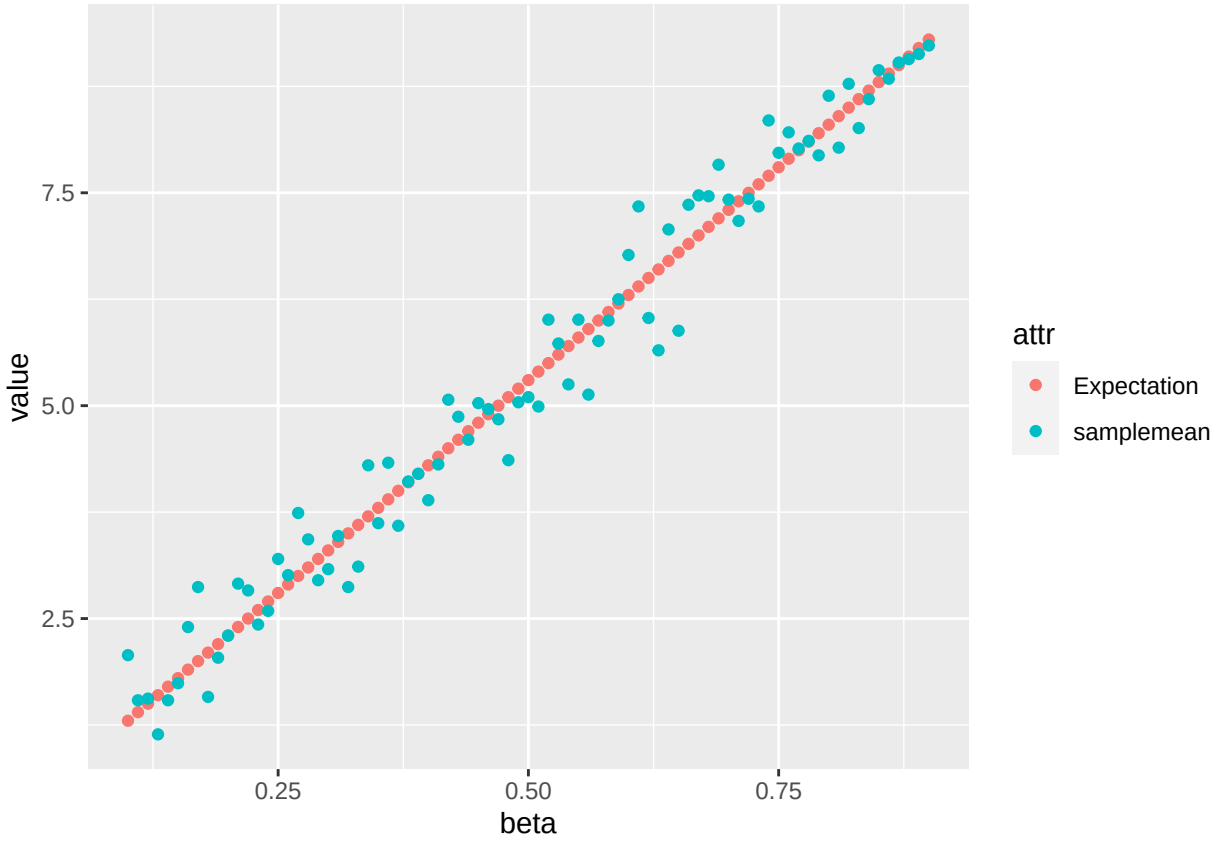
```
## [1] 8.25
```

1.1 (Q6)

```
temp <- data.frame(beta = seq(1/10, 9/10, 0.01))



myData <- temp %>%
  mutate(sample_x = map(.x = beta, ~sample_x_0310(1/10, .x, 100)),
         samplemean = map_dbl(.x = sample_x, ~mean(.x)))
newEx <- c()
for (i in myData$beta) {
  vals <- c(3, 10, 0)
  probs <- c(1/10, i, 1 - (0.1 + i))
  Ex <- weighted.mean(vals, probs)
  newEx <- c(newEx, Ex)
}

myData <- myData %>%
  mutate(Expectation = newEx)
```

1.1 (Q7)

```
d <- gather(myData, key = "attr", value = "value", Expectation, samplemean)

ggplot(d, aes(x = beta, y = value, color = attr)) +
  geom_point()
```



1.2 (Q1)

To prove that p_λ, we need to integrate the given function. If the integration boild down to 1, then we can say that the given function is well-defined probability density function.

$\int_0^\infty \lambda e^{-\lambda x}\, dx$
Let $u = -\lambda x, so du = -\lambda dx$

$\Rightarrow \int_0^\infty \lambda e^{-\lambda x}\, dx = -\int_0^\infty e^{-\lambda x}(-\lambda\, dx) = \int_0^\infty e^u\, du$

The integral of e^e is itself.
$\Rightarrow -\int_0^\infty e^u\, du$
$\Rightarrow -e^u + C = -e^{-\lambda x} + C$
$\Rightarrow -[e^{-\lambda x}] - [-e^0]$

$\Rightarrow 1$
$\therefore$ given function is a well-defined probability density function.

Cumulative Distribution Function
For $x > 0$, we have
$F_X(x) = \int_0^x \lambda e^{\lambda t} dt = 1 - e^{-\lambda x}$
So we can express CDF as
$F_X(x) = (1 - e^{-\lambda x})u(x)$

Quantile Function
The cumulative distribution function of the exponential distribution is:

$$F_X(x) = \begin{cases} 0\,, & \text{if } x < 0 \\ 1 - \exp[-\lambda x]\,, & \text{if } x \geq 0\,. \end{cases}$$

The quantile function $Q_X(p)$ is defined as the smallest x, such that $F_X(x) = p$

$$Q_X(p) = \min\{x \in \mathbb{R} \,|\, F_X(x) = p\}$$

Thus, we have $Q_X(p) = -\infty$, if $p = 0$. When $p > 0$, it holds that,

$$Q_X(p) = F_X^{-1}(x)$$

$$\begin{aligned} p &= 1 - \exp[-\lambda x] \\ \exp[-\lambda x] &= 1 - p \\ -\lambda x &= \ln(1-p) \\ x &= -\frac{\ln(1-p)}{\lambda} \end{aligned}$$

1.2 (Q2)

```
my_cdf_exp <- function(x, lambda) {
  if (x < 0) {
    return(0)
  } else {
    val = 1 - exp(-lambda * x)
    return(val)
  }
}

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
```

```
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
```

1.2 (Q3)

```
my_quantile_exp <- function(p, lambda) {
  ans = -log(1 - p) / lambda
  return(ans)
}

lambda <- 1/2
map_dbl(.x=seq(0.01, 0.99, 0.2), .f=~my_cdf_exp(x=.x,lambda=lambda))
```

```
## [1] 0.004987521 0.099675477 0.185352684 0.262876626 0.333023189
```

```
test_inputs <- seq(0.01, 0.99, 0.2)
my_quantile_output <- map_dbl(.x = test_inputs, .f = ~my_quantile_exp(p =.x, lambda = lambda))
inbuilt_quantile_output <- map_dbl(.x = test_inputs, .f = ~qexp(p = .x, rate = lambda))
all.equal(my_quantile_output, inbuilt_quantile_output)
```

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

1.2 (Q4)

Expectation of exponential distribution
Let X $Exponential(\lambda)$. We can find its expected value as follows, using integration by parts:

$$
\begin{aligned}
EX &= \int_0^\infty x\lambda e^{-\lambda x} dx \\
Consideringy &= \lambda x \\
&= \frac{1}{\lambda}\int_0^\infty y e^{-y} dy \\
&= \frac{1}{\lambda}\Big[-e^{-y} - ye^{-y} \Big]_0^\infty \\
&= \frac{1}{\lambda}
\end{aligned}
$$

Variance of exponential distribution
We have,

$$
\begin{aligned}
EX^2 &= \int_0^\infty x^2\lambda e^{-\lambda x} dx \\
&= \frac{1}{\lambda^2}\int_0^\infty y^2 e^{-y} dy
\end{aligned}
$$

$$= \frac{1}{\lambda^2}\Big[-2e^{-y} - 2ye^{-y} - y^2e^{-y} \Big]_0^\infty$$

$$= \frac{2}{\lambda^2}.$$

Therfore, we get

$$\text{Var}(X) = EX^2 - (EX)^2 = \frac{2}{\lambda^2} - \frac{1}{\lambda^2} = \frac{1}{\lambda^2}.$$

1.3 The Binomial distribution and the central limit theorem
1.3 (Q1)
In Binomial distribution, the probability of seeing k successes in n trials where the probability of success in each trial is p (and $q = 1 - p$) is given by,

$$P(X = k) = ({}_nC_k)p^k q^{n-k}$$

We know that,

$$E(X + Y) = E(X) + E(Y) \qquad \text{and} \qquad Var(X + Y) = Var(X) + Var(Y)$$

Let, X_t be a random variable that gives the number of successes seen a single trial (i.e., either 0 or 1). The distribution of X_t is simple in the extreme:

x	0	1
$P(X_t = x)$	q	p

Which gives,

$$E(X_t) = (0)(q) + (1)(p) = p$$

and

$$\begin{aligned} Var(X_t) &= \big[(0^2)(q) + (1^2)(p)\big] - p^2 \\ &= p - p^2 \\ &= p(1 - p) \\ &= pq \end{aligned}$$

Now, returning to the expected value of the original random variable X that follows a binomial distribution, note that

$$\begin{aligned} E(X) &= \underbrace{E(X_t) + E(X_t) + \cdots E(X_t)}_{\text{n terms}} \\ &= nE(X_t) \\ &= np \end{aligned}$$

Finding the variance of X is just as immediate:

$$\begin{aligned} Var(X) &= \underbrace{Var(X_t) + Var(X_t) + \cdots Var(X_t)}_{\text{n terms}} \\ &= nVar(X_t) \\ &= npq \end{aligned}$$

This, of course, immediately gives the standard deviation of X:

$$SD(X) = \sqrt{Var(X)} = \sqrt{npq}$$

1.3 (Q2)

```
pmf <- c()
x <- seq(0, 50, 1)

for (i in x) {
  binom_temp <- dbinom(i, size = 50, prob = 7/10)
  pmf <- c(pmf, binom_temp)
}

binom_df <- data.frame(x, pmf)
head(binom_df)
```

```
##   x          pmf
## 1 0 7.178980e-27
## 2 1 8.375477e-25
## 3 2 4.787981e-23
## 4 3 1.787513e-21
## 5 4 4.900764e-20
## 6 5 1.052031e-18
```

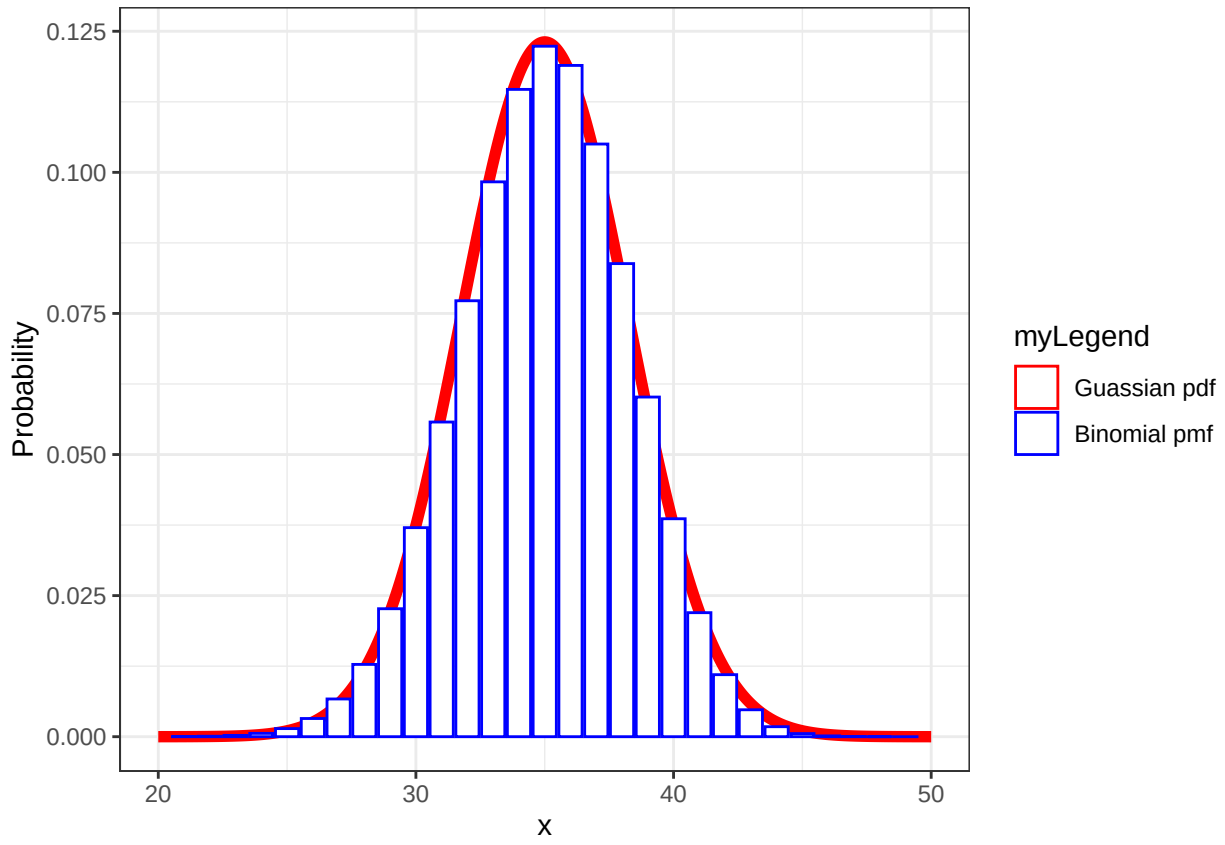
1.3 (Q3)

```
x <- seq(0, 50, 0.01)
pdf <- c()

for(i in  x) {
  dnorm_temp <- dnorm(i, mean = (50 * 0.7), sd = sqrt(50 * 0.7 * (1 - 0.7)))
  pdf <- c(pdf, dnorm_temp)
}

guassian_df <- data.frame(x, pdf)
head(guassian_df)
```

```
##      x          pdf
## 1 0.00 5.707825e-27
## 2 0.01 5.901264e-27
## 3 0.02 6.101201e-27
## 4 0.03 6.307852e-27
## 5 0.04 6.521440e-27
## 6 0.05 6.742196e-27
```

1.3 (Q4)

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

ggplot() + labs(x = "x", y = "Probability") + theme_bw() +
  geom_line(data = guassian_df, aes(x, y = pdf, color = "Guassian pdf"), size = 2) +
  geom_col(data = binom_df, aes(x = x, y = pmf, color = "Binomial pmf", fill = "Binomial pmf")) +
  scale_color_manual(name = "myLegend", values = colors) +
  scale_fill_manual(name = "myLegend", values = fill) +
  xlim(c(20, 50))
```



1.4 The Guassian Distribution

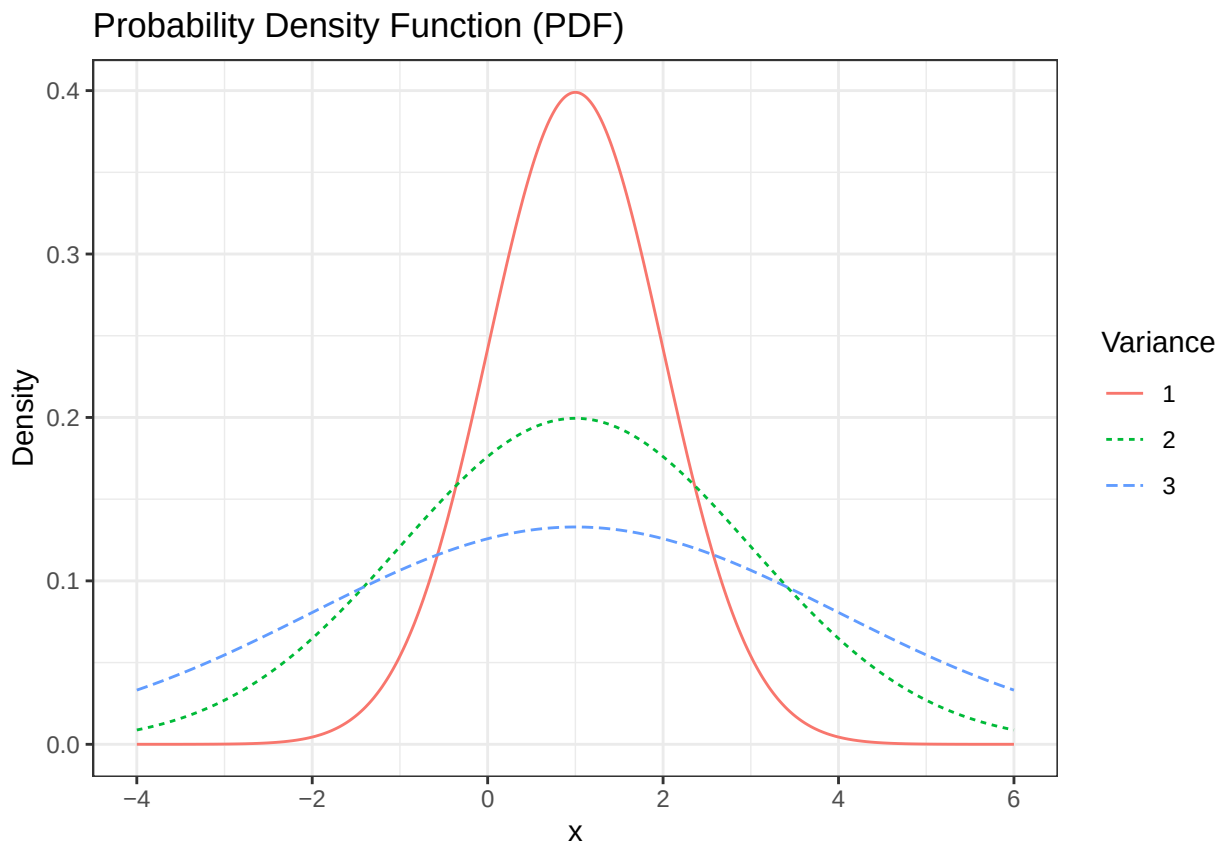
1.4 (Q1)

```
x <- seq(-4, 6, 0.01)
pdf1 <- c()
pdf2 <- c()
pdf3 <- c()
```

```r
for (i in x) {
  dnorm_temp1 <- dnorm(i, mean = 1, sd = 1)
  pdf1 <- c(pdf1, dnorm_temp1)

  dnorm_temp2 <- dnorm(i, mean = 1, sd = 2)
  pdf2 <- c(pdf2, dnorm_temp2)

  dnorm_temp3 <- dnorm(i, mean = 1, sd = 3)
  pdf3 <- c(pdf3, dnorm_temp3)
}
df <- data.frame(x, pdf1, pdf2, pdf3)

colnames(df) <- c("x", "1", "2", "3")

d <- df %>%
  pivot_longer(c(`1`, `2`, `3`), names_to = "Variance", values_to = "Density")
head(d)
```

```
## # A tibble: 6 x 3
##       x Variance    Density
##   <dbl> <chr>         <dbl>
## 1 -4    1        0.00000149
## 2 -4    2        0.00876
## 3 -4    3        0.0332
## 4 -3.99 1        0.00000156
## 5 -3.99 2        0.00887
## 6 -3.99 3        0.0333
```

```r
ggplot(d, aes(x = x, y = Density, color = Variance, linetype = Variance)) +
  geom_line() +
  theme_bw() +
  labs(title = "Probability Density Function (PDF)")
```



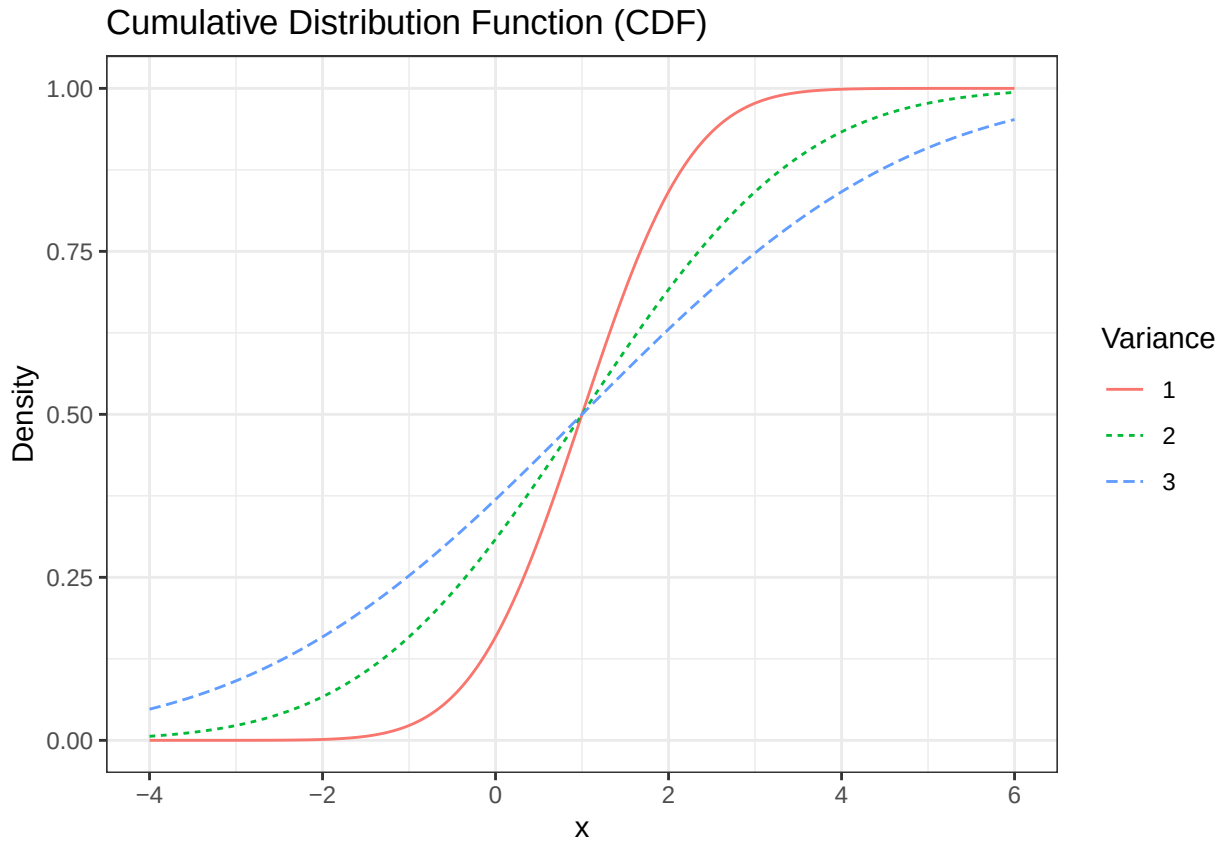
1.4 (Q2)

```r
x <- seq(-4, 6, 0.01)

cdf1 <- c()
cdf2 <- c()
cdf3 <- c()


for (i in x) {
  pnorm_temp1 <- pnorm(i, mean = 1, sd = 1)
  cdf1 <- c(cdf1, pnorm_temp1)

  pnorm_temp2 <- pnorm(i, mean = 1, sd = 2)
  cdf2 <- c(cdf2, pnorm_temp2)

  pnorm_temp3 <- pnorm(i, mean = 1, sd = 3)
  cdf3 <- c(cdf3, pnorm_temp3)
}
df <- data.frame(x, cdf1, cdf2, cdf3)

colnames(df) <- c("x", "1", "2", "3")
```

```r
d <- df %>%
  pivot_longer(c(`1`, `2`, `3`), names_to = "Variance", values_to = "Density")
head(d)
```

```
## # A tibble: 6 x 3
##       x Variance     Density
##   <dbl> <chr>          <dbl>
## 1 -4    1        0.000000287
## 2 -4    2        0.00621
## 3 -4    3        0.0478
## 4 -3.99 1        0.000000302
## 5 -3.99 2        0.00630
## 6 -3.99 3        0.0481
```

```r
ggplot(d, aes(x = x, y = Density, color = Variance, linetype = Variance)) +
  geom_line() +
  theme_bw() +
  labs(title = "Cumulative Distribution Function (CDF)")
```

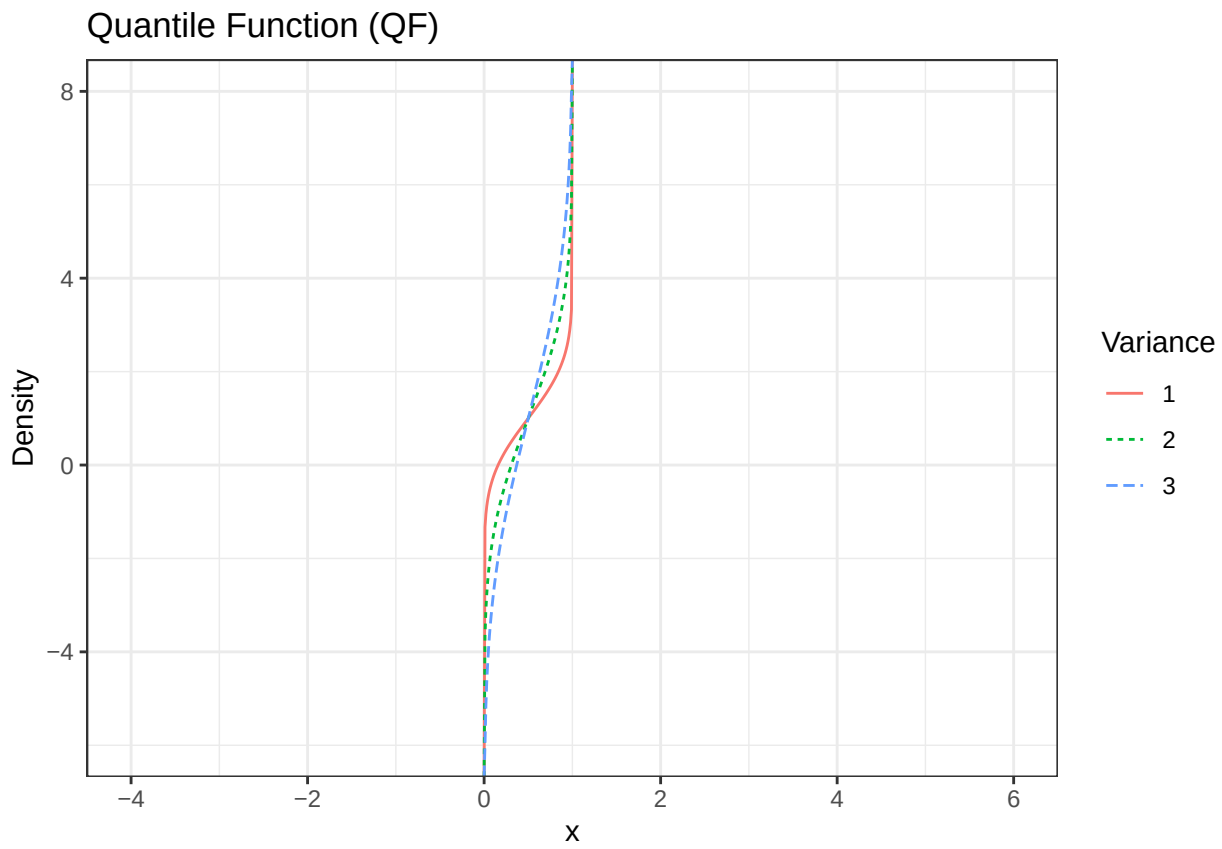


1.4 (Q3)

```r
x <- seq(-4, 6, 0.01)

qdf1 <- c()
qdf2 <- c()
qdf3 <- c()


for (i in x) {
  qnorm_temp1 <- qnorm(i, mean = 1, sd = 1)
  qdf1 <- c(qdf1, qnorm_temp1)

  qnorm_temp2 <- qnorm(i, mean = 1, sd = 2)
  qdf2 <- c(qdf2, qnorm_temp2)

  qnorm_temp3 <- qnorm(i, mean = 1, sd = 3)
  qdf3 <- c(qdf3, qnorm_temp3)
}
df <- data.frame(x, qdf1, qdf2, qdf3)

colnames(df) <- c("x", "1", "2", "3")

d <- df %>%
  pivot_longer(c(`1`, `2`, `3`), names_to = "Variance", values_to = "Density")
head(d)
```

```
## # A tibble: 6 x 3
##       x Variance Density
##   <dbl> <chr>      <dbl>
## 1 -4    1            NaN
## 2 -4    2            NaN
## 3 -4    3            NaN
## 4 -3.99 1            NaN
## 5 -3.99 2            NaN
## 6 -3.99 3            NaN
```

```r
ggplot(d, aes(x = x, y = Density, color = Variance, linetype = Variance)) +
  geom_line() +
  theme_bw() +
  labs(title = "Quantile Function (QF)")
```



1.4 (Q4)

```
set.seed(42)
standardGaussianSample <- rnorm(100, 0, 1)
head(standardGaussianSample)
```

```
## [1]  1.3709584 -0.5646982  0.3631284  0.6328626  0.4042683 -0.1061245
```

1.4 (Q5)

```
# Alpha = 3 and Beta = 1
mean1Var3GaussianSampleA <- (3 * standardGaussianSample) + 1
head(mean1Var3GaussianSampleA)
```

```
## [1]  5.1128753 -0.6940945  2.0893852  2.8985878  2.2128050  0.6816265
```

$\alpha = 3$ and $\beta = 1$.

1.4 (Q6)

```
set.seed(42)
mean1Var3GaussianSampleB <- rnorm(100, 1, 3)
head(mean1Var3GaussianSampleB)
```

```
## [1]  5.1128753 -0.6940945  2.0893852  2.8985878  2.2128050  0.6816265
```

```
all.equal(mean1Var3GaussianSampleA, mean1Var3GaussianSampleB)
```

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

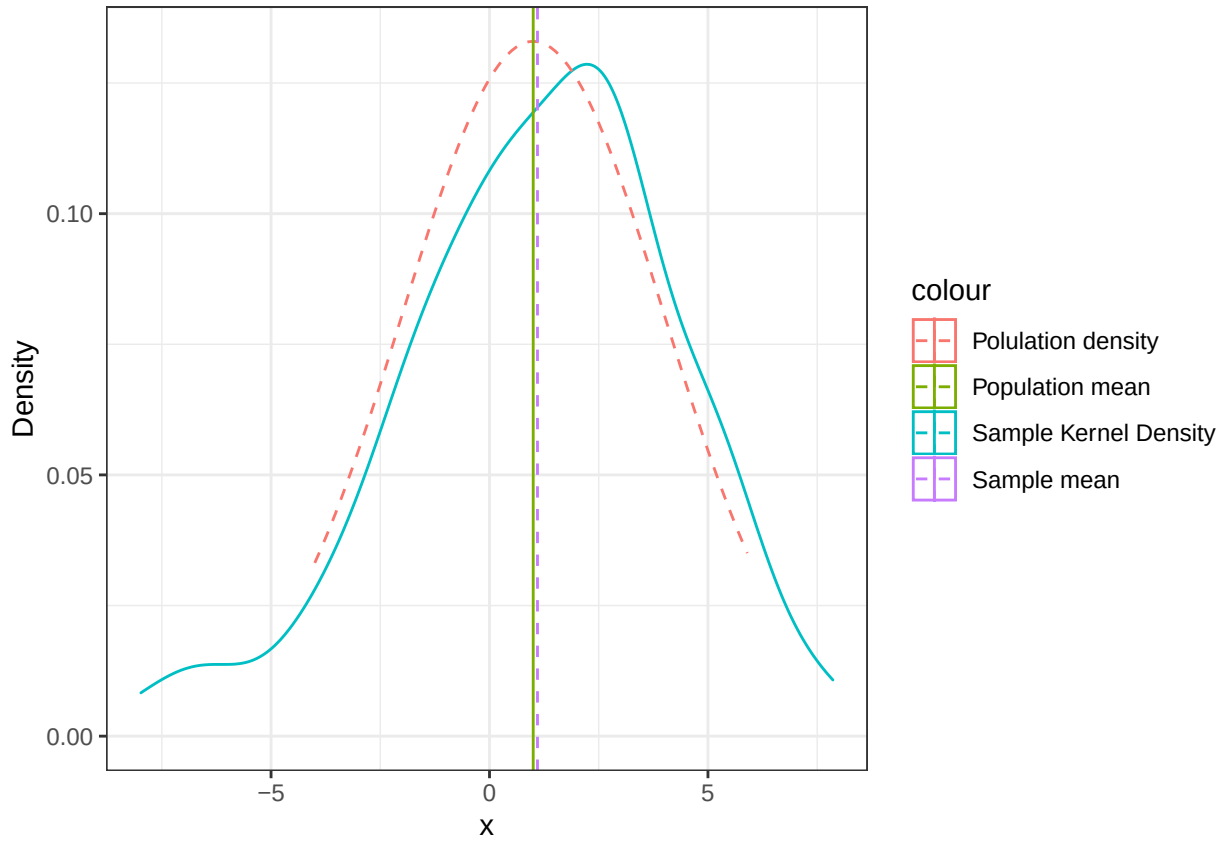
We can see that, entries of $mean1Var3GaussianSampleA$ and $mean1Var3GaussianSampleB$ are the same.

1.4 (Q7)

```
x <- seq(-4, 5.99, 0.1)
pdf <- c()
for (i in x) {
  dnorm_t <- dnorm(i, mean = 1, sd = 3)
  pdf <- c(pdf, dnorm_t)
}

dfMean1Var3 <- data.frame(x, mean1Var3GaussianSampleA, pdf)
head(dfMean1Var3)
```

```
##      x mean1Var3GaussianSampleA        pdf
## 1 -4.0                5.1128753 0.03315905
## 2 -3.9               -0.6940945 0.03503388
## 3 -3.8                2.0893852 0.03697361
## 4 -3.7                2.8985878 0.03897741
## 5 -3.6                2.2128050 0.04104417
## 6 -3.5                0.6816265 0.04317253
```

```
colors <- c("Polulation density" = "red", "Sample mean" = "green", "Sample Kernel Density" = "blue", "P
fill<-c("Gaussianpdf" = "white", "Binomialpmf" = "white")


ggplot(dfMean1Var3, aes(mean1Var3GaussianSampleA)) +
  geom_density(aes(color = "Sample Kernel Density")) +
  geom_line(aes(x = x, y = pdf, color = "Polulation density"), linetype = 2) +
  geom_vline(aes(xintercept = mean(mean1Var3GaussianSampleA), color = "Sample mean"), linetype = 2) +
  geom_vline(aes(xintercept = 1, color = "Population mean")) +
  xlab("x") + ylab("Density") +
  theme_bw()
```



1.4 (Q8)

We can obtain any normal random varibale by shifting and scaling a standard normal random variable. In particular, define

$$X = \sigma Z + \mu, \qquad \text{where } \sigma > 0$$

Then,

$$EX = \sigma EZ + \mu = \mu,$$
$$\text{Var}(X) = \sigma^2 \text{Var}(Z) = \sigma^2$$

We say that, X is a normal random variable with mean μ and variance σ^2. We write $X \sim N(\mu, \sigma^2)$.

Conversely, if $X \sim N(\mu, \sigma^2)$, the random variable defined by $Z = \frac{X-\mu}{\sigma}$ is a standard normal random variable, i.e., $Z \sim N(0,1)$. To find the CDF of $X \sim N(\mu, \sigma^2)$, we can write,

$$F_X(x) = P(X \leq x)$$
$$= P(\sigma Z + \mu \leq x) \qquad (\text{where } Z \sim N(0,1))$$
$$= P\left(Z \leq \frac{x-\mu}{\sigma}\right)$$

$$= \Phi\left(\frac{x-\mu}{\sigma}\right)$$

To find the PDF, we can take th derivative of F_X,

$$\begin{aligned} f_X(x) &= \frac{d}{dx} F_X(x) \\ &= \frac{d}{dx} \Phi\left(\frac{x-\mu}{\sigma}\right) \\ &= \frac{1}{\sigma} \Phi'\left(\frac{x-\mu}{\sigma}\right) \qquad \text{(chain rule for derivative)} \\ &= \frac{1}{\sigma} f_Z\left(\frac{x-\mu}{\sigma}\right) \\ &= \frac{1}{\sigma\sqrt{2\pi}} \exp\left\{-\frac{(x-\mu)^2}{2\sigma^2}\right\} \end{aligned}$$

2. Location estimators with Gaussian data

```
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

simulation_df <- crossing(trial = seq(num_trials_per_sample_size),
                          sample_size = seq(min_sample_size, max_sample_size, sample_size_inc)) %>%
  mutate(simulation = pmap(.l = list(trial, sample_size),
                           .f = ~rnorm(.y, mean = mu_0, sd = sigma_0))) %>%
  mutate(sample_md = map_dbl(.x = simulation, .f = median))  %>%
  group_by(sample_size) %>%
  summarise(msq_error_md = mean((sample_md - mu_0) ^ 2))
```

2 (Q1)

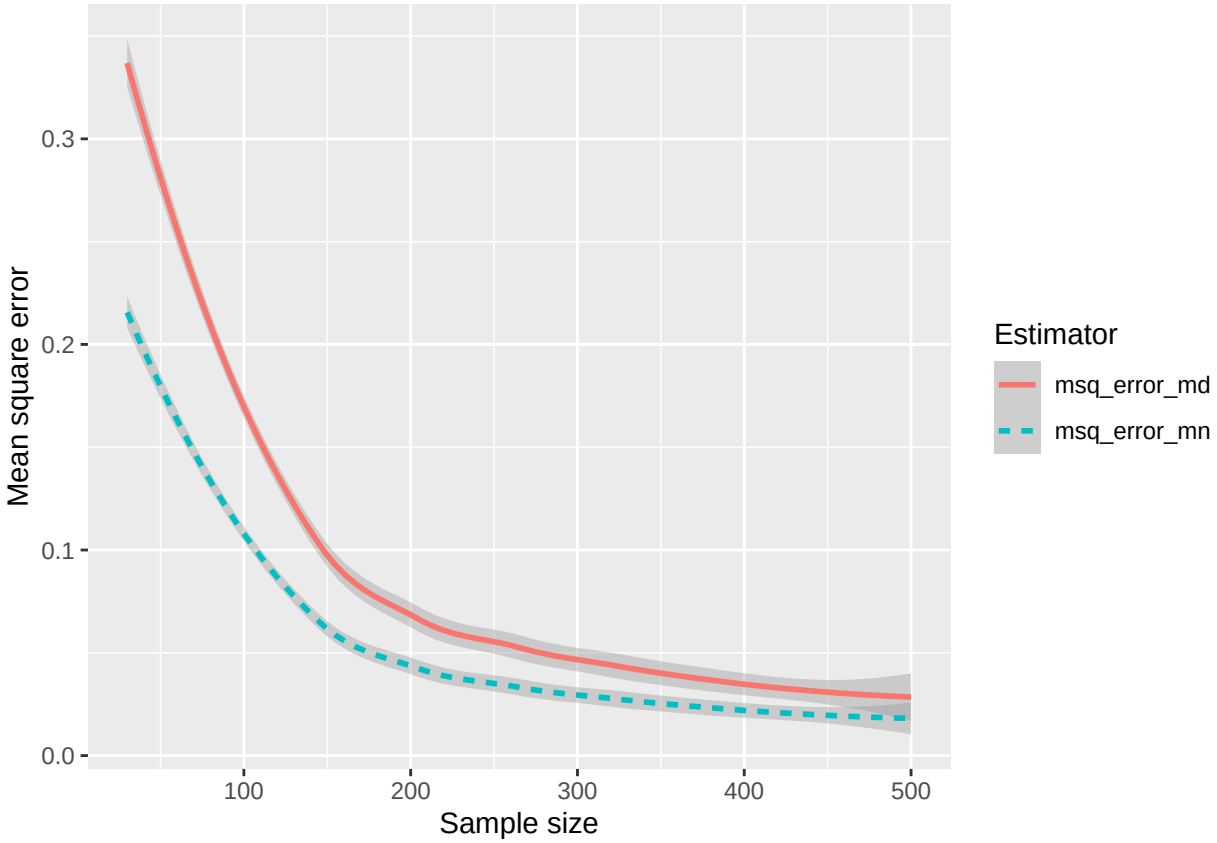
The population median of a Gaussian random variable is **0.9997083**.

2 (Q2)

```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

simulation_df2 <- crossing(trial = seq(num_trials_per_sample_size),
                           sample_size = seq(min_sample_size, max_sample_size, sample_size_inc)) %>%
  mutate(simulation = pmap(.l = list(trial, sample_size),
                           .f = ~rnorm(.y, mean = mu_0, sd = sigma_0))) %>%
  mutate(sample_md = map_dbl(.x = simulation, .f = median),
         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))
```

```r
d <- simulation_df2 %>%
  pivot_longer(!sample_size, names_to = "Estimator", values_to = "Mean square error")

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

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



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

3 (Q1)

Let's assume that $Var(X) = \sigma^2$ is finite. In this case, we can use Chebyshev's inequality to write,

$$P(|\overline{X} - \mu| \geq \epsilon) \leq \frac{\text{Var}(\overline{X})}{\epsilon^2}$$
$$= \frac{\text{Var}(X)}{n\epsilon^2}$$

Which goes to zero as $n \to \infty$.