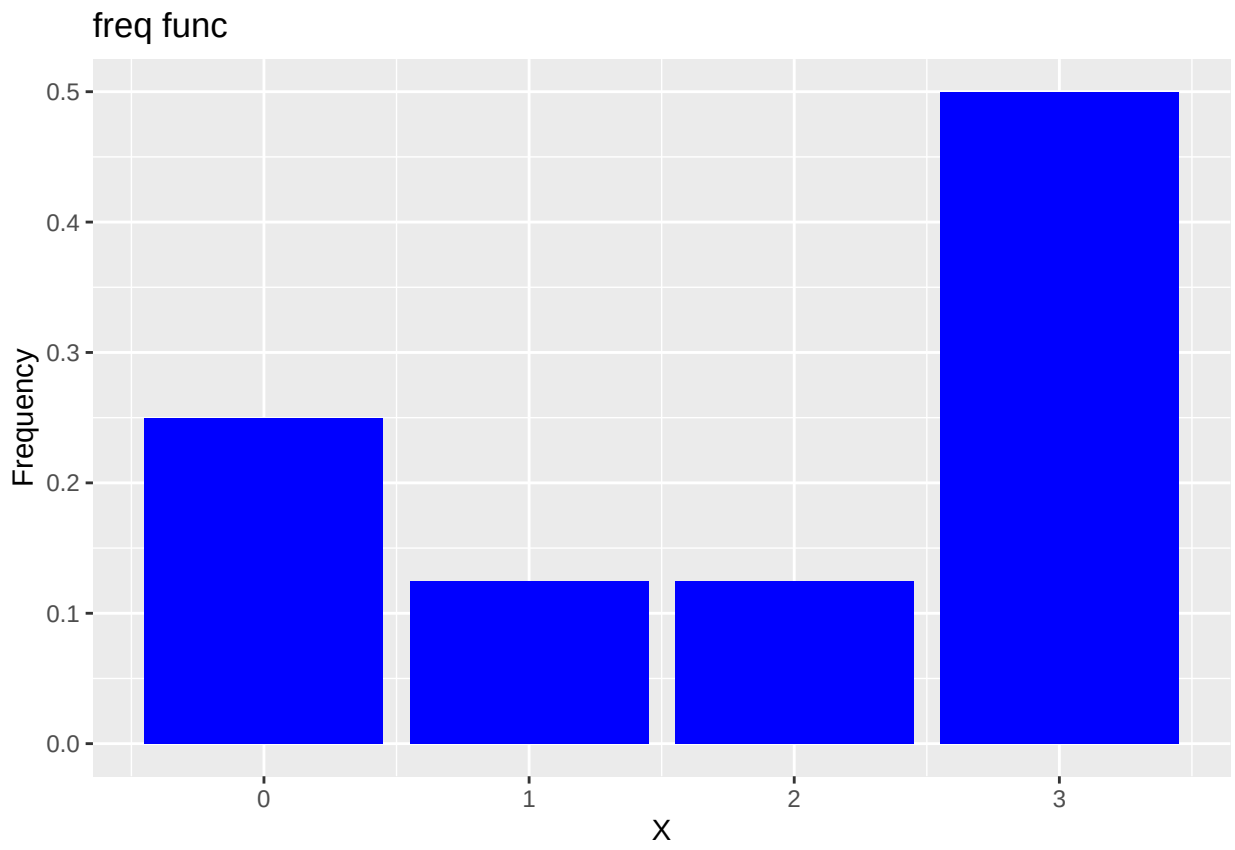
Shivam Bajaj BMI-510

Q1

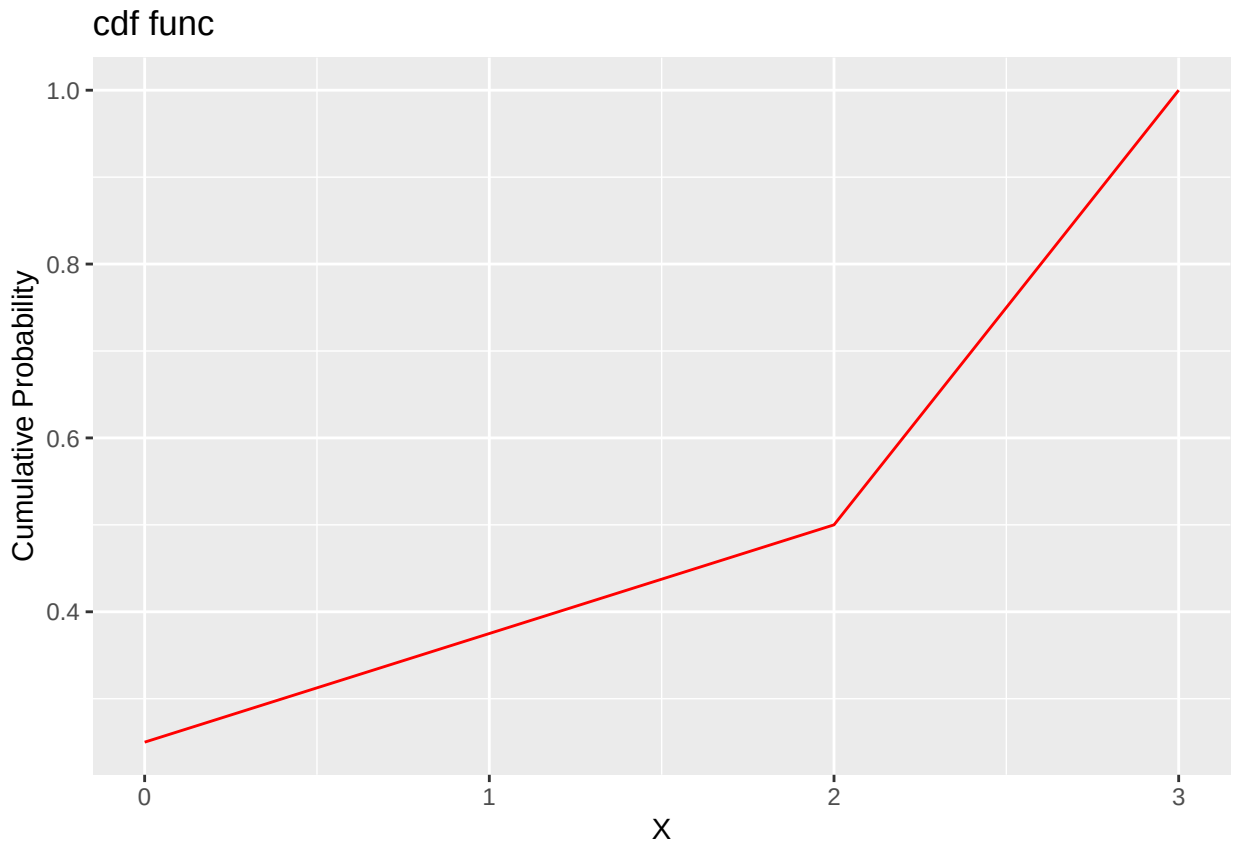
```
library(ggplot2)
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.2 --
## v tibble  3.1.8      v dplyr   1.0.10
## v tidyr   1.2.1      v stringr 1.5.0
## v readr   2.1.3      v forcats 0.5.2
## v purrr   1.0.1
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
df <- data.frame(x <- c(0,1,2,3) , prob <- c(0.25,0.125,0.125,0.5))

#frequency plot
ggplot(df, aes(x, prob)) +
  geom_bar(stat = "identity", fill = "blue") +
  labs(x = "X", y = "Frequency")+ggtitle("freq func")
```

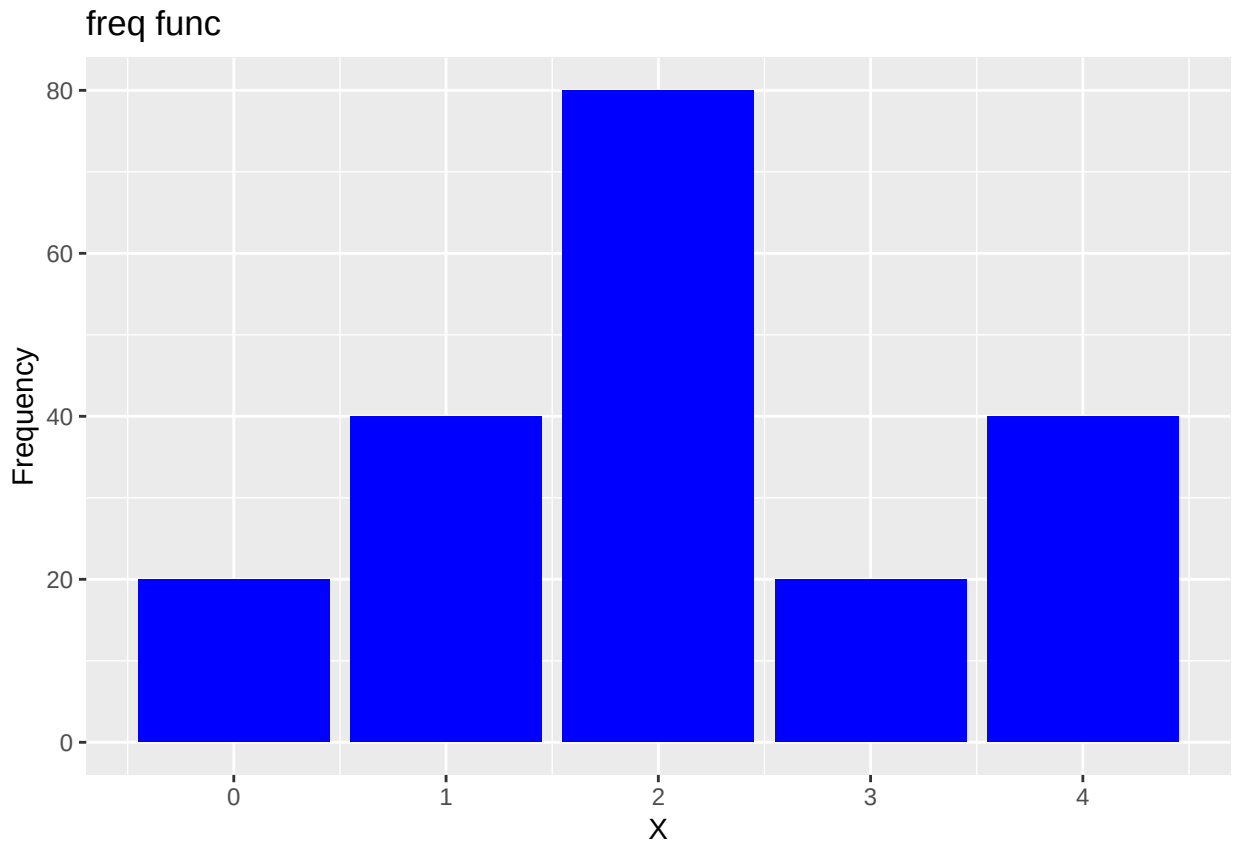


```r
#cumulative distribution function plot
ggplot(df, aes(x, cumsum(prob))) +
  geom_line(color = "red") +
  labs(x = "X", y = "Cumulative Probability")+ggtitle("cdf func")
```



Q2

```r
cumulative_frequency <- c(0, 20, 60, 140, 160, 200)
df <- data.frame(x <- c(0,1,2,3,4) , frequency <- diff(cumulative_frequency))

ggplot(df, aes(x, frequency)) +
  geom_bar(stat = "identity", fill = "blue") +
  labs(x = "X", y = "Frequency")+ ggtitle("freq func")
```



Q3

```
pi_user <- 3.14159

##(i)
identical(pi_user, pi)
```

```
## [1] FALSE
```

```
##(ii)
all.equal(pi_user, pi)
```

```
## [1] "Mean relative difference: 8.446646e-07"
```

```
##(iii)
all.equal(pi_user, pi, tolerance = 1e-5)
```

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

```
##identical - tests for exact and complete match

##checks relative difference , or how close exactly they are

##By specifying tolerance, the function will return true if the difference in both the terms are less t
```

Q4

```
a <- 3
x <- 1:12

match(a,x) == min(which(x==a))
```

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

```
x %in% 4
```

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

```
##checks if each number is 4 or not

x %in% c(5,10)
```

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

```
##checks if each number is either a 5 or 10
```

Q5

```
BoxMuller <- function(n) {

  u <- runif(n)
  v <- runif(n)

  z1 <- rep(0,n)
  z2 <- rep(0,n)

  for (i in 1:n){
    z1[i] <- sqrt(-2 * log(u[i]))* cos(2*pi*v[i])
    z2[i] <- sqrt(-2 * log(u[i]))* sin(2*pi*v[i])
  }

   return(cbind(z1, z2))

  #return(z1,z2)

}

samples <- BoxMuller(500)
##mean(BoxMuller(500))
##var(BoxMuller(500))

print("mean = nearly 0")
```

```
## [1] "mean = nearly 0"
```

```r
mean(samples[, 1])
```

```
## [1] -0.008358501
```

```r
print("var = nearly 1")
```

```
## [1] "var = nearly 1"
```

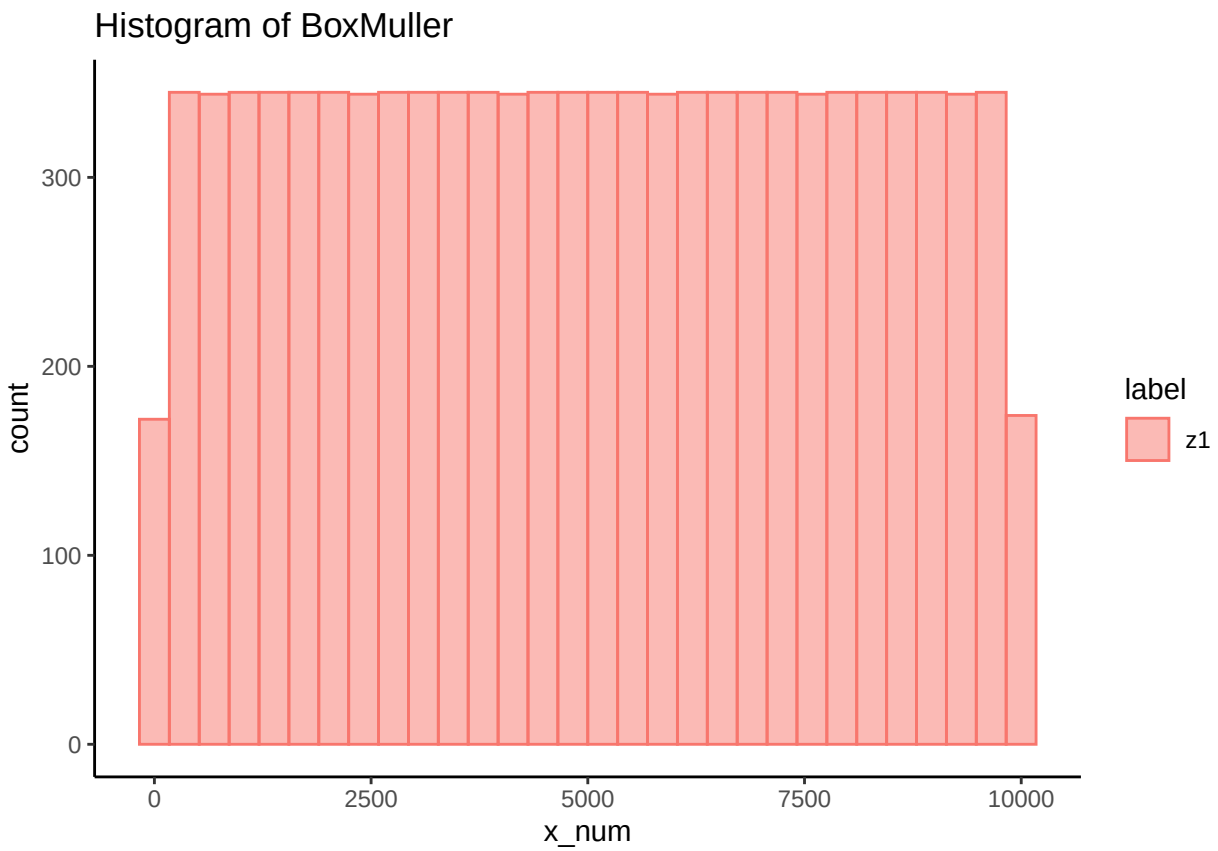
```r
var(samples[, 1])
```

```
## [1] 0.9094444
```

Q6

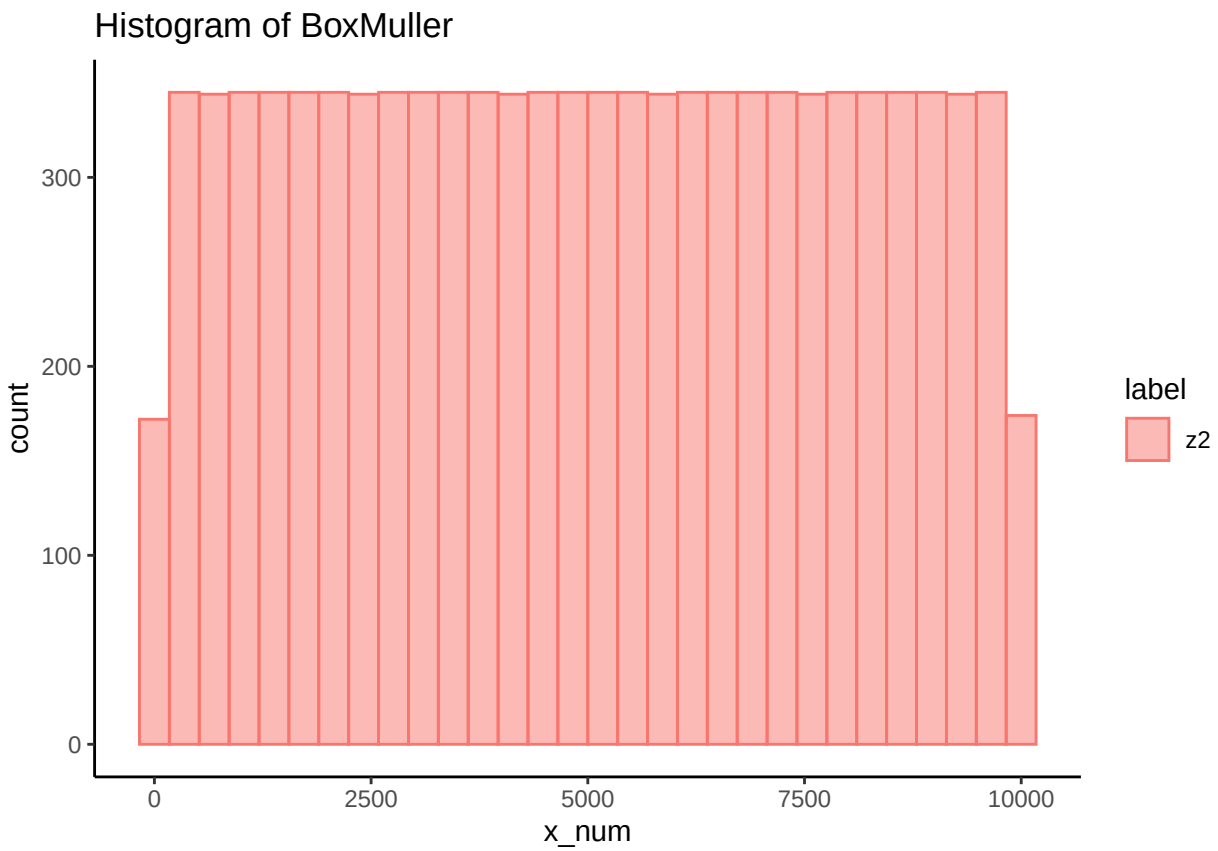
```r
samples <- data.frame(s <- BoxMuller(10000))

label = rep("z1",10000)
df5 <- as.data.frame(cbind(sa <- samples[, "z1"],label))
label = rep("z2",10000)
df6 <- as.data.frame(sa <- cbind(samples[, "z2"],label))

samples <- rnorm(10000)
label <- rep("rnorm", 10000)
df2 <- as.data.frame(sa <- cbind(samples,label))

colnames(df5) <- c("sample", "label")
colnames(df6) <- c("sample", "label")
colnames(df2) <- c("sample", "label")

df3 <- rbind(df5,df2,df6)
df3$x_num <- as.numeric(factor(df3$sample))
df5$x_num <- as.numeric(factor(df5$sample))
df6$x_num <- as.numeric(factor(df6$sample))
df2$x_num <- as.numeric(factor(df2$sample))

ggplot(df5, aes(x=x_num, color=label, fill=label)) +
  geom_histogram(position = "identity", alpha = 0.5) +
  theme(legend.position = "top")+
  theme_classic()+
  ggtitle("Histogram of BoxMuller")
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



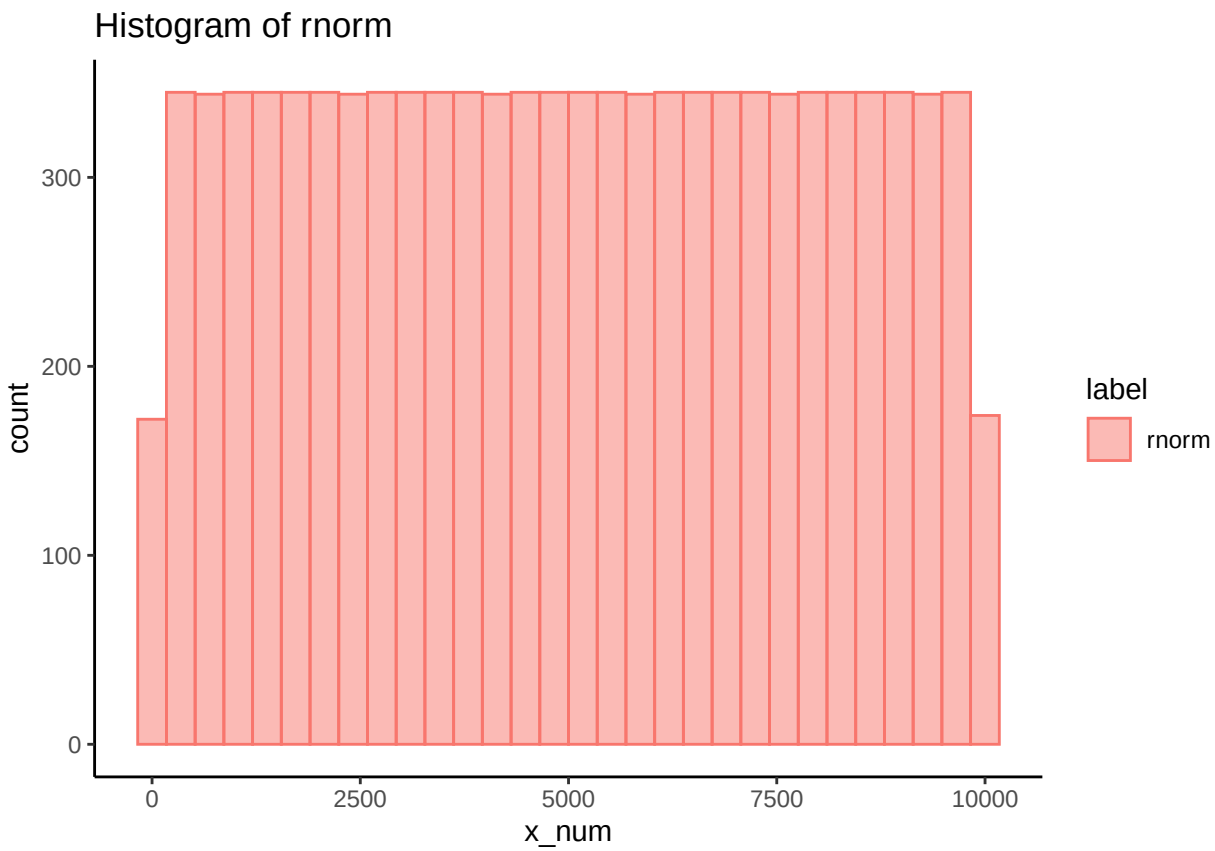

```
ggplot(df6, aes(x=x_num, color=label, fill=label)) +
  geom_histogram(position = "identity", alpha = 0.5) +
  theme(legend.position = "top")+
  theme_classic()+
  ggtitle("Histogram of BoxMuller")
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



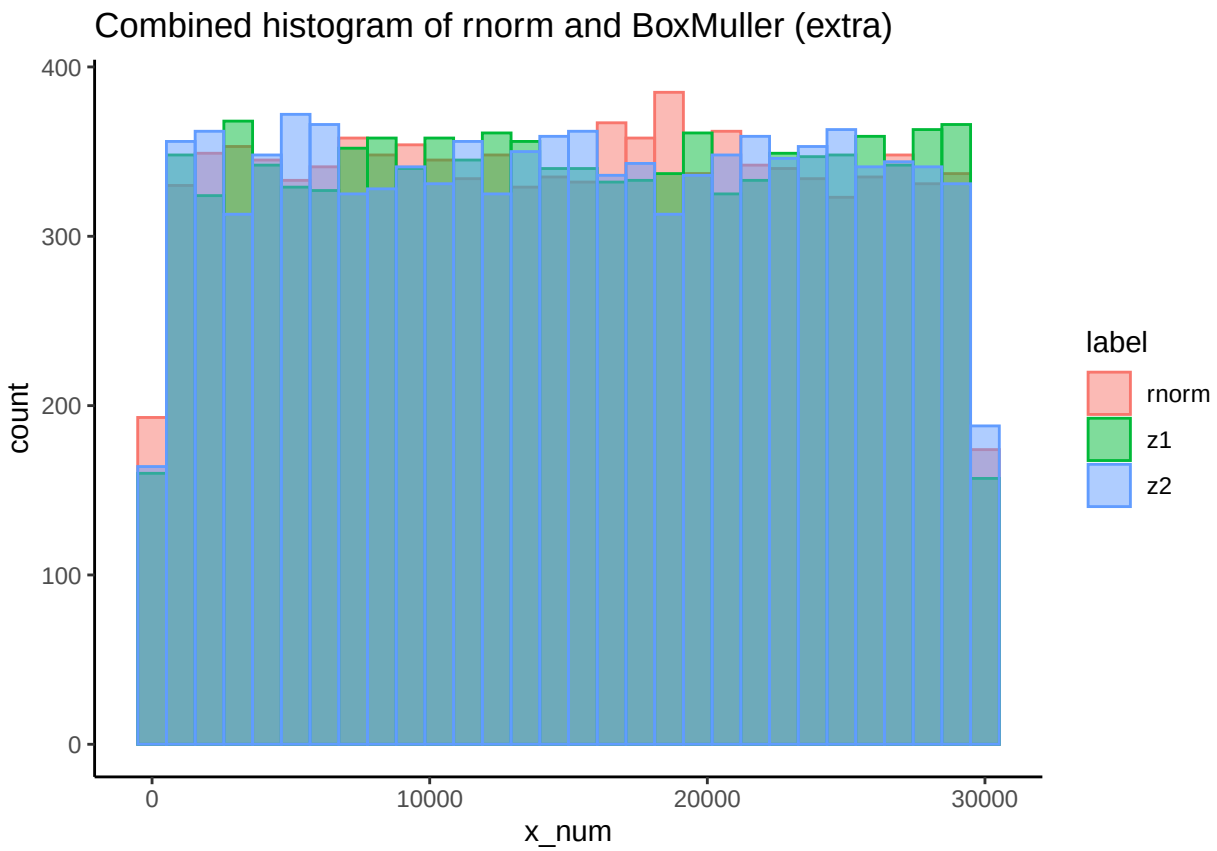

```
ggplot(df2, aes(x=x_num, color=label, fill=label)) +
  geom_histogram(position = "identity", alpha = 0.5) +
  theme(legend.position = "top")+
  theme_classic()+
  ggtitle("Histogram of rnorm")
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



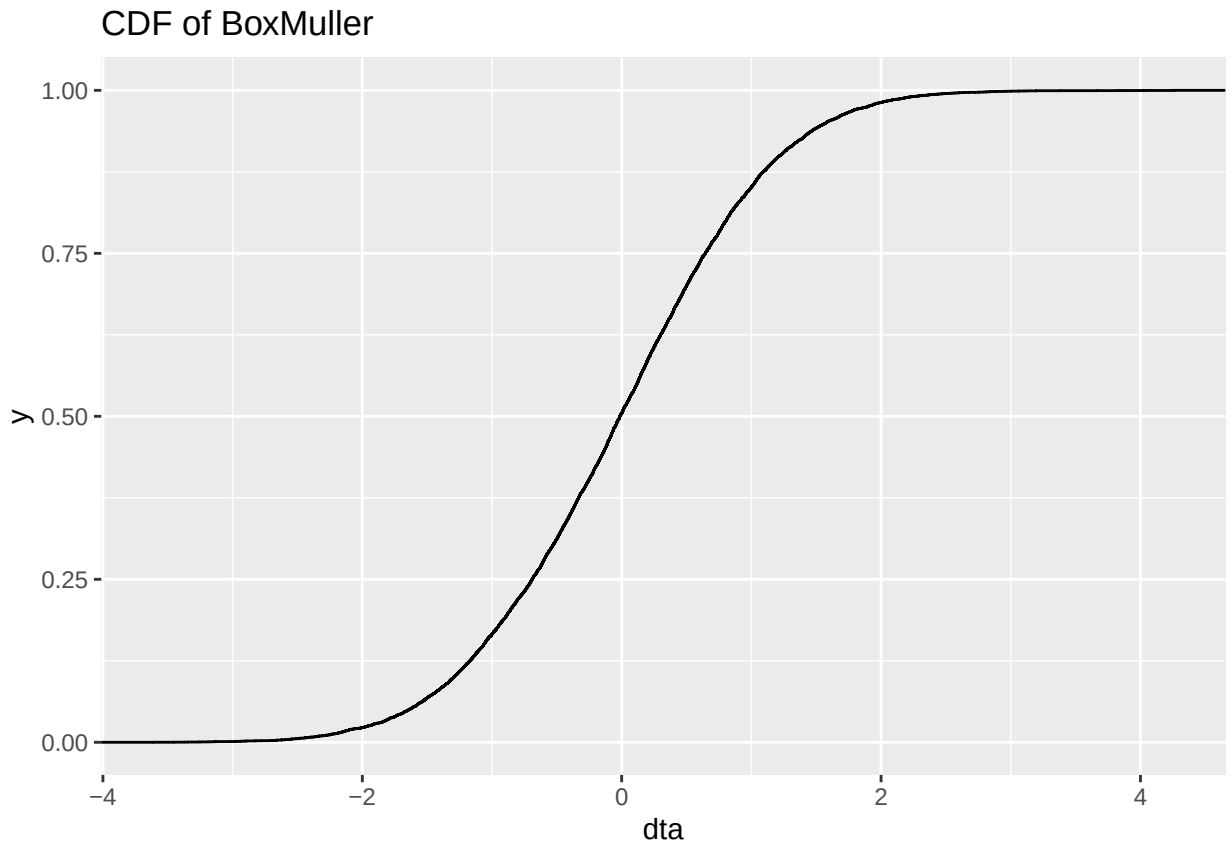

```
ggplot(df3, aes(x=x_num, color=label, fill=label)) +
  geom_histogram(position = "identity", alpha = 0.5) +
  theme(legend.position = "top")+
  theme_classic()+
  ggtitle("Combined histogram of rnorm and BoxMuller (extra)")
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



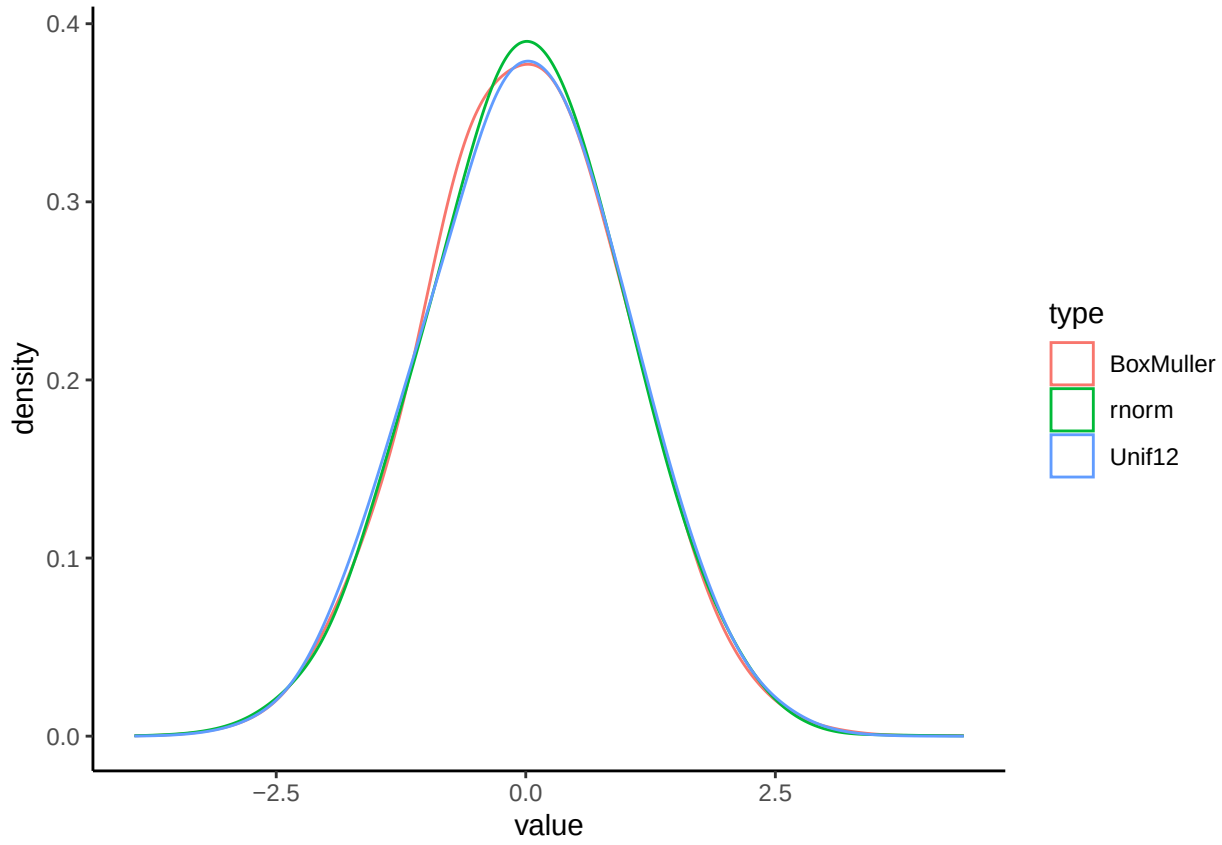
Q7

```
sample <- BoxMuller(10000)
df <- data.frame(dta <- sample[, "z1"])

ggplot(df, aes(dta))+
  stat_ecdf(geom = "step")+
  ggtitle("CDF of BoxMuller")
```



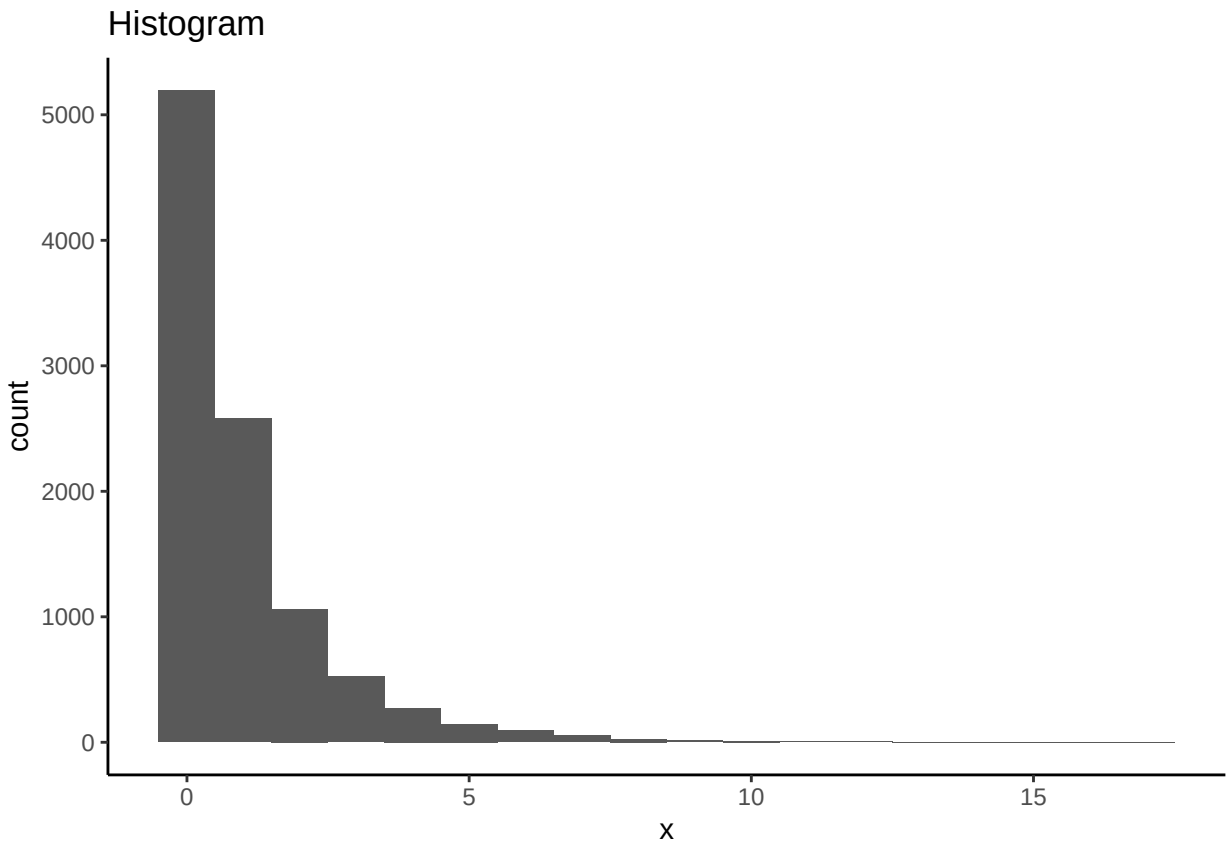
Q8

```
Unif12 <- function(n) {
  x <- runif(n * 12, min = -0.5, max = 0.5)
  return(rowSums(matrix(x, n, 12)))
}

n <-10000

data_rnorm <- rnorm(n)

sample <- BoxMuller(n)

data_BoxMuller <-  sample[, "z1"]


data_Unif12 <- Unif12(n)



df43 <- data.frame(value = c(data_rnorm, data_BoxMuller, data_Unif12),
                   type = c(rep("rnorm", n), rep("BoxMuller", n), rep("Unif12", n)))

ggplot(df43, aes(value, color = type)) +
  geom_density(adjust = 2) +
  theme_classic()
```

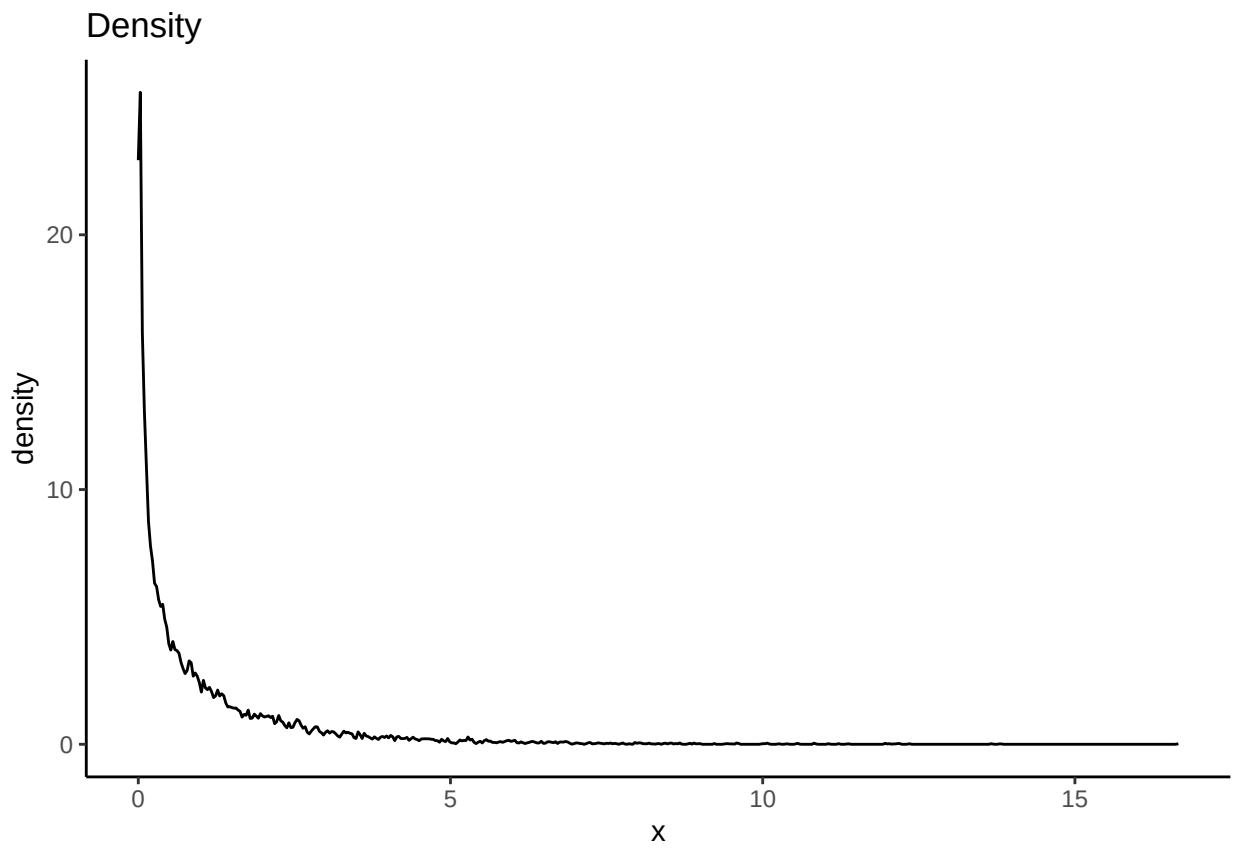


Q9

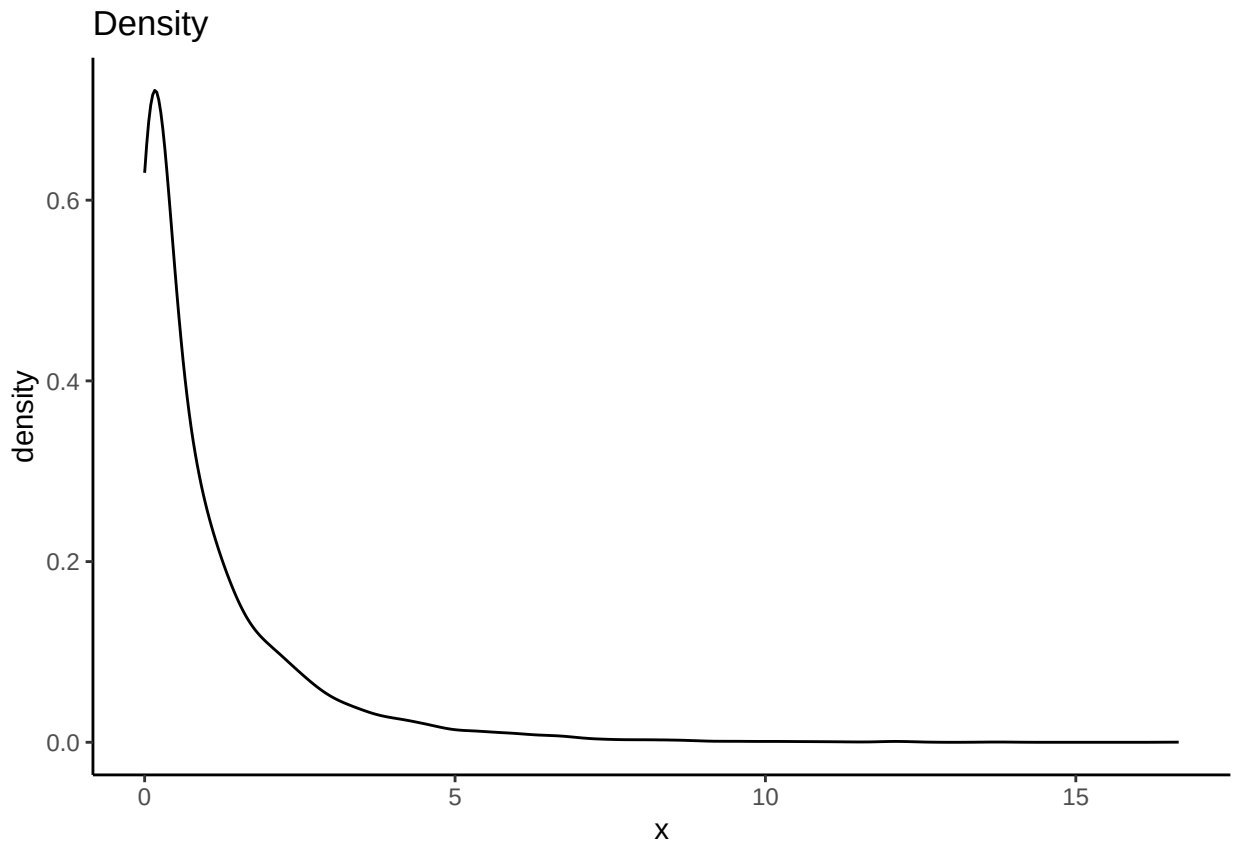
```
df <- data.frame(x <- rchisq(10000, df = 1))

ggplot(df, aes(x))+
  geom_histogram(binwidth = 1) +
  theme_classic()+
  ggtitle("Histogram")
```




```
ggplot(df,aes(x)) +
  geom_density(adjust = 0.01) +
  theme_classic()+
  ggtitle("Density")
```



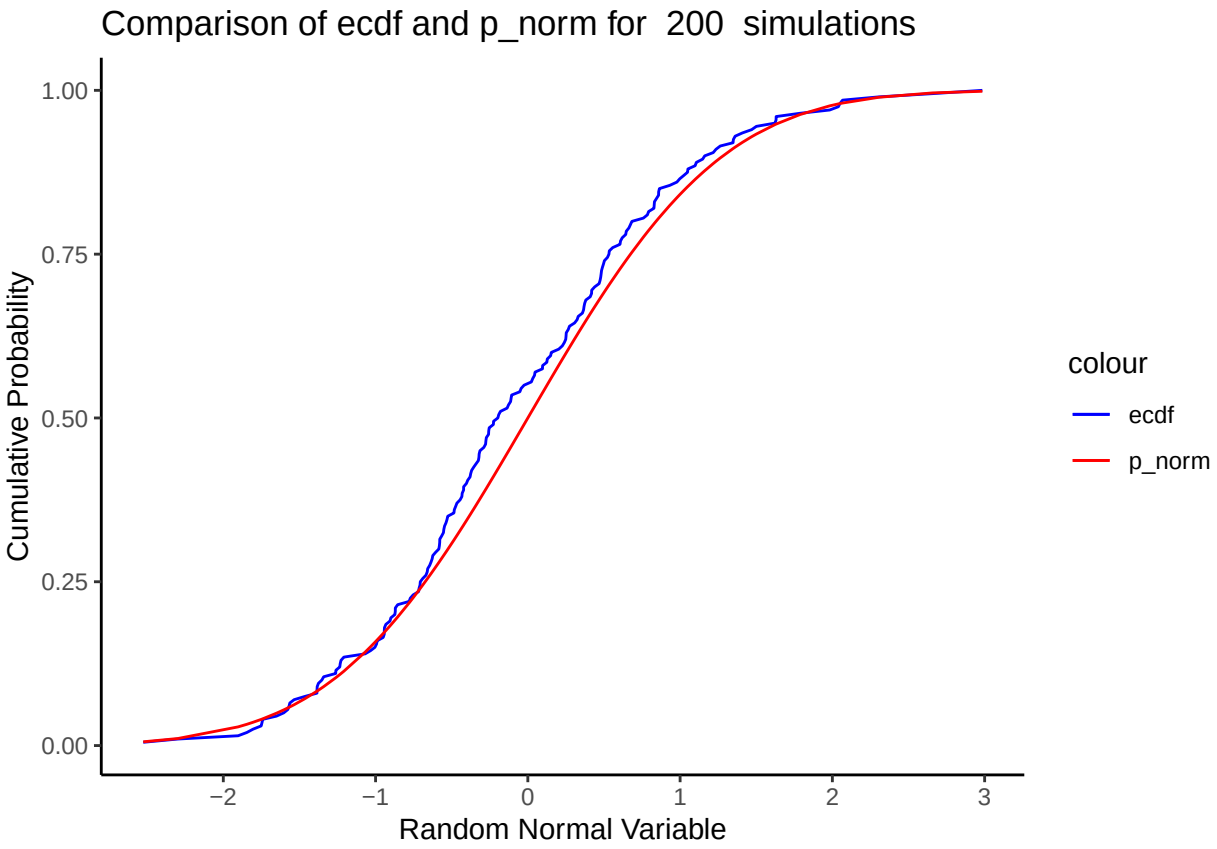

```
ggplot(df,aes(x)) +
  geom_density(adjust = 2) +
  theme_classic()+
  ggtitle("Density")
```



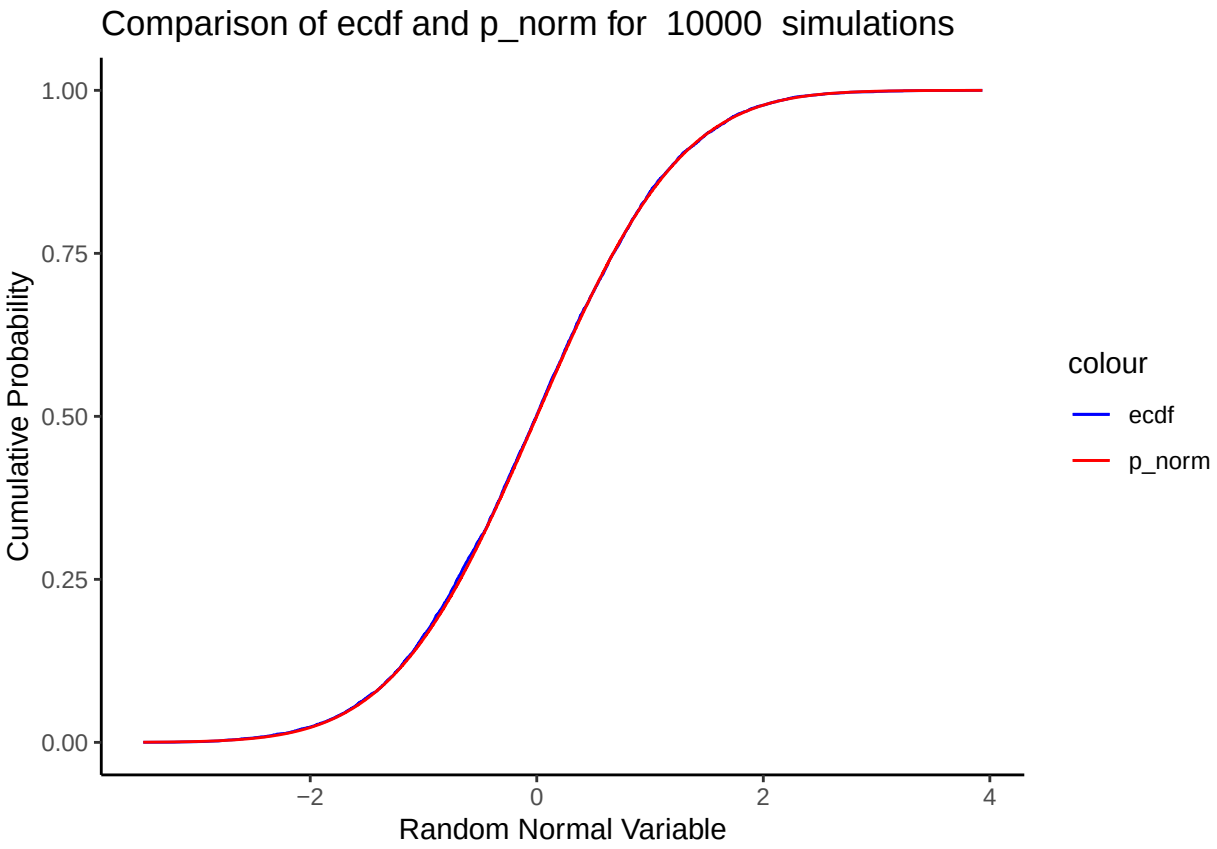

```
##Lower limit of the histogram determine the smallest value in the data. What is noticed widely is maxm

##Adjust in geom_density helps in smoothening the plot. A high value of adjust means more smoother plot
```

Q10

```
library(stats)

ecdf_and_pnorm_df_calculator <- function(n){

x <- rnorm(n)

df <- data.frame(x = sort(x),
                 y_ecdf = ecdf(x)(sort(x)),
                 y_pnorm = pnorm(sort(x)))


ggplot(df, aes(x)) +
  geom_line(aes(y = y_ecdf, color = "ecdf")) +
  geom_line(aes(y = y_pnorm, color = "p_norm")) +
  xlab("Random Normal Variable") + ylab("Cumulative Probability") +
  scale_color_manual(values = c("ecdf" = "blue", "p_norm" = "red")) +
  theme_classic()+  ggtitle(paste("Comparison of ecdf and p_norm for ",n," simulations"))

}
ecdf_and_pnorm_df_calculator(200)
```




```
ecdf_and_pnorm_df_calculator(10000)
```




```
#With 10,000 samples, the ECDF and pnorm will be even closer to each other than when compared to 200 si
```