Statistical Calculation and Software

Assignment 1

Hanbin Liu 11912410

1.1

(a)

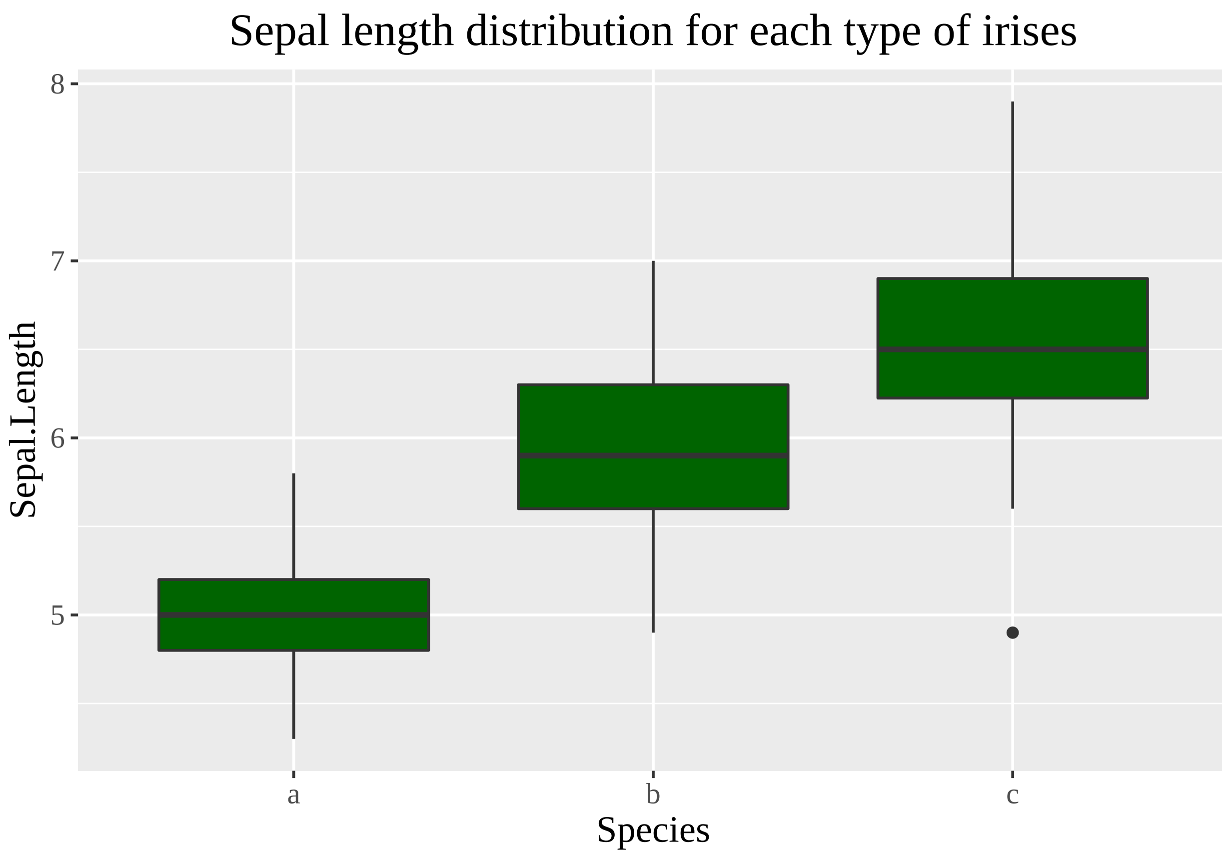
```
data('iris')
```

```
Species <- factor(iris$Species, labels = c("a", "b", "c"))
iris <- data.frame(iris[, 1:4], Species)
table(iris$Species)
```

```
##
##  a  b  c
## 50 50 50
```

(b)

```
ggplot(iris, aes(x = Species, y = Sepal.Length)) +
  geom_boxplot(fill = "darkgreen") +
  ggtitle("Sepal length distribution for each type of irises") +
  theme(plot.title = element_text(hjust = 0.5), text = element_text(size = 14, family = "serif"))
```

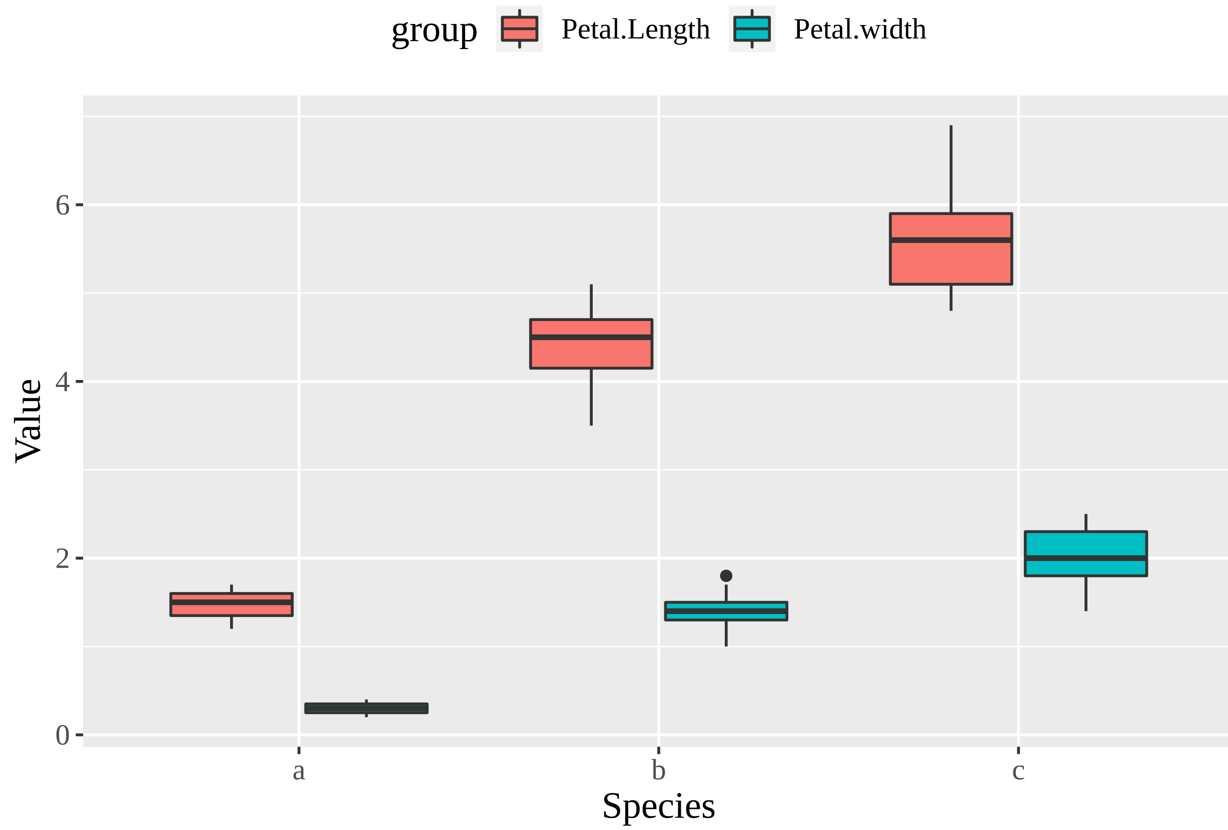


(c)

```r
ind <- which(iris["Sepal.Length"] > 5.5)
iris_ed <- iris[ind, ]

length <- iris_ed$Petal.Length
width <- iris_ed$Petal.Width
value <- c(length, width)
group <- c(rep('Petal.Length', length(width)), rep('Petal.width', length(width)))
species <- c(iris_ed$Species, iris_ed$Species)
data <- data.frame(species, group, value)

ggplot(data, aes(x = species, y = value, fill = group)) +
  geom_boxplot() + xlab("Species") + ylab("Value") +
  theme(legend.position = "top", text = element_text(size = 14, family = "serif"))
```



1.2

```
F2000 <- read.csv("F2000.csv")
```

(a)&(b)

```
skewness <- function(x){
  xixi <- x - mean(x)
  skewness <- sum(xixi^3) / ((length(x) - 1) * var(x)^1.5)
  return(skewness)
}

x <- F2000[, "marketvalue"]
skewness(x)
```

```
## [1] 6.430443
```

```
skewness(log(x))
```

```
## [1] 0.03204195
```

```r
skewness(1 - x^-1)
```

```
## [1] -19.92494
```

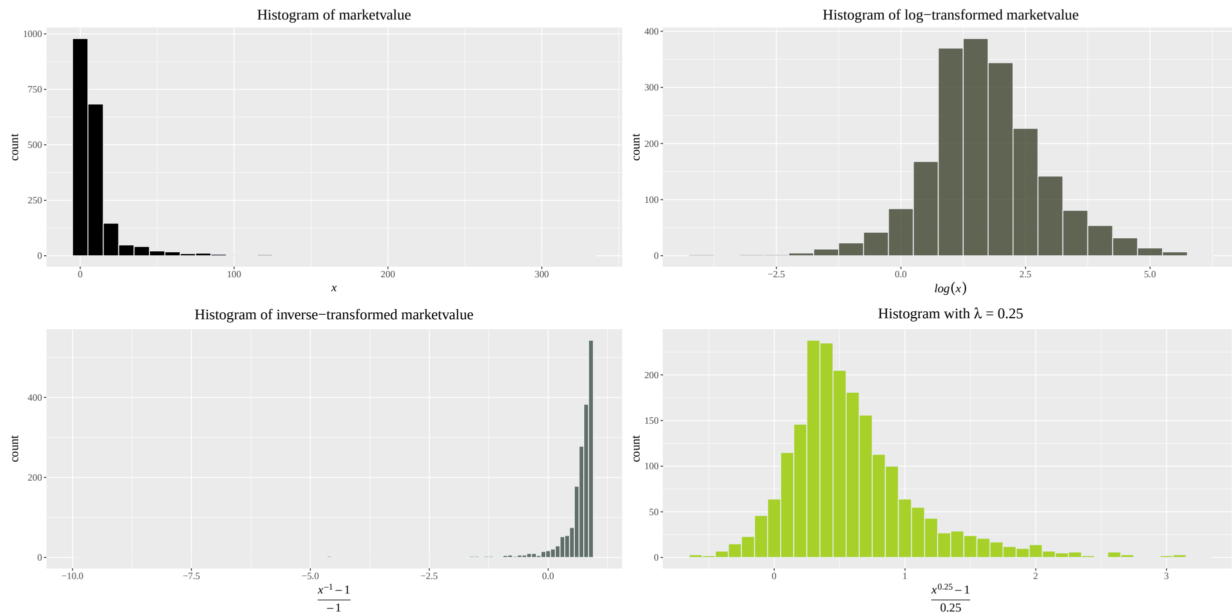
```r
skewness((x^0.25 - 1)/0.25)
```

```
## [1] 1.381235
```

(c)

```r
library(patchwork)
p1 <- ggplot(F2000, aes( x = marketvalue)) +
  geom_histogram(binwidth = 10, fill = "#000000", color = "#e9ecef") +
  ggtitle("Histogram of marketvalue") +
  theme(plot.title = element_text(hjust = 0.5), text = element_text(size = 14, family = "serif")) +
  xlab(expression(italic(x)))
p2 <- ggplot(F2000, aes( x = log(marketvalue))) +
  geom_histogram(binwidth = 0.5, fill = "#3E432E", color = "#e9ecef", alpha = 0.8) +
  ggtitle("Histogram of log-transformed marketvalue") +
  theme(plot.title = element_text(hjust = 0.5), text = element_text(size = 14, family = "serif")) +
  xlab(expression(italic(log(x))))
p3 <- ggplot(F2000, aes( x = 1 - (marketvalue)^-1)) +
  geom_histogram(binwidth = 0.1, fill = "#616F69", color = "#e9ecef") +
  ggtitle("Histogram of inverse-transformed marketvalue") +
  theme(plot.title = element_text(hjust = 0.5), text = element_text(size = 14, family = "serif")) +
  xlim(c(-10, 1)) +
  xlab(expression(italic(frac(x^-1 - 1, -1))))
p4 <- ggplot(F2000, aes( x = marketvalue^0.25 - 1)) +
  geom_histogram(binwidth = 0.1, fill = "#A7D129", color = "#e9ecef") +
  ggtitle(expression(paste('Histogram with ', italic(lambda), " = 0.25"))) +
  theme(plot.title = element_text(hjust = 0.5), text = element_text(size = 14, family = "serif")) +
  xlab(expression(italic(frac(x^0.25 - 1, 0.25))))

p1 + p2 + p3 + p4
```



(d)

```r
ind <- which(is.na(F2000[, "profits"]))
F2000[, "name"][ind]
```

```
## [1] "AMP"                    "HHG"                        "NTL"
## [4] "US Airways Group"       "Laidlaw International"
```

```r
fivenum(F2000[, "sales"][ind])
```

```
## [1] 3.50 4.48 5.40 5.50 5.68
```

(e)

```r
# number of different countries
length(table(F2000$country))
```

```
## [1] 61
```

```r
xixi <- data.frame(table(F2000$country))
mean <- 1:61; median <- mean
for(i in 1:61){
  ind <- which(F2000$country == xixi[i, 1])
  mean[i] <- mean(F2000[ind, ]$assets)
  median[i] <- median(F2000[ind, ]$assets)
}

countries <- data.frame(country = xixi[, 1], num_of_companies = xixi[, 2],
                        mean_assets = mean, median_assets = median)
write.table(countries, file = "countries.txt")
```

(f)

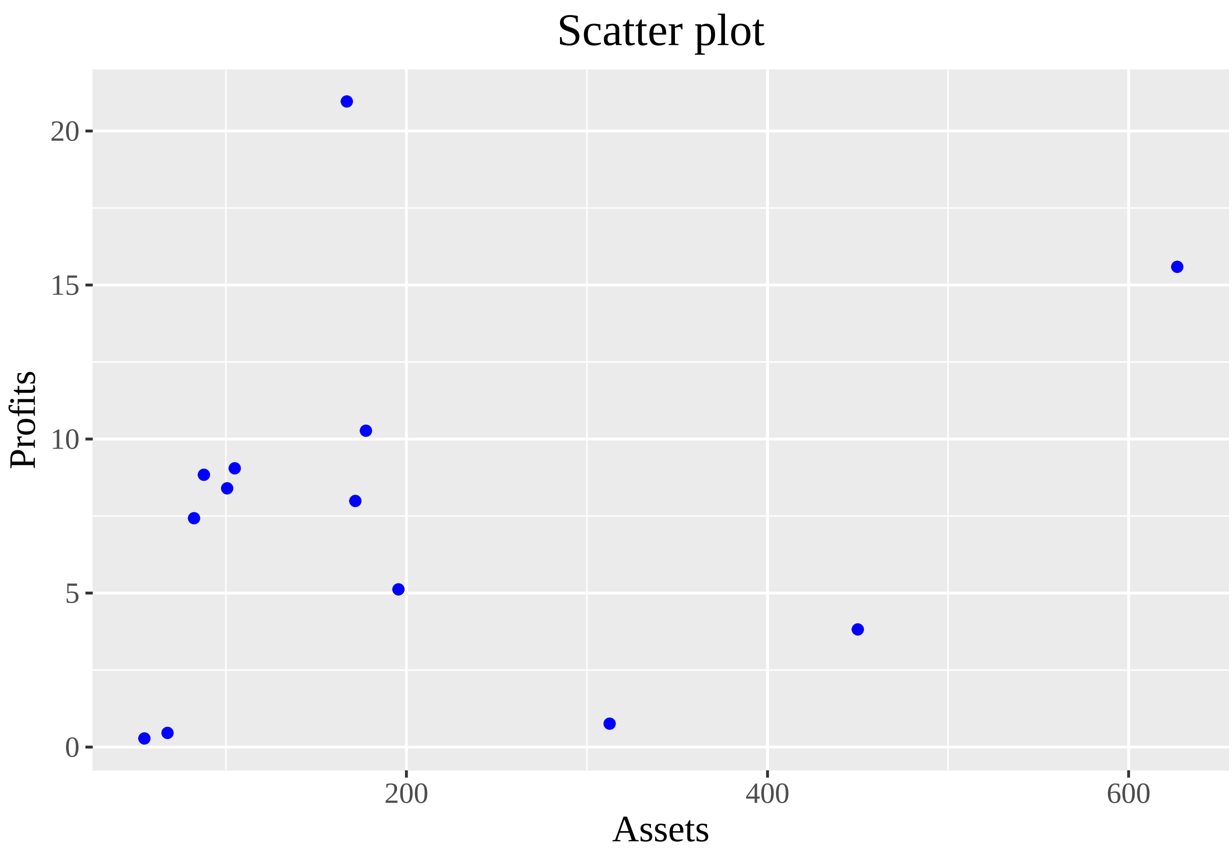
```r
# 2 ways
way_1 <- F2000[F2000$sales > 100, ]
way_2 <- F2000[which(F2000$sales > 100), ]
# sort
selected_data <- way_1[-c(3, 4, 8)]
ind_1 <- order(selected_data$sales, decreasing = TRUE)
selected_data[ind_1, ]
```

```
##     rank                   name  sales profits assets
## 10    10         Wal-Mart Stores 256.33    9.05 104.91
## 5      5                      BP 232.57   10.27 177.57
## 4      4               ExxonMobil 222.88   20.96 166.99
## 29    29          General Motors 185.52    3.82 450.00
## 75    75              Ford Motor 164.20    0.76 312.56
## 21    21         DaimlerChrysler 157.13    5.12 195.58
## 8      8            Toyota Motor 135.82    7.99 171.71
## 2      2        General Electric 134.19   15.59 626.93
## 13    13 Royal Dutch/Shell Group 133.50    8.40 100.72
## 17    17                   Total 131.64    8.84  87.84
## 23    23           ChevronTexaco 112.94    7.43  82.36
## 156  156              Mitsubishi 112.76    0.46  67.69
## 225  225             Mitsui & Co 111.98    0.28  54.88
```

```r
ind_2 <- order(selected_data$assets)
selected_data[ind_2, ]
```

```
##     rank                   name  sales profits assets
## 225  225             Mitsui & Co 111.98    0.28  54.88
## 156  156              Mitsubishi 112.76    0.46  67.69
## 23    23           ChevronTexaco 112.94    7.43  82.36
## 17    17                   Total 131.64    8.84  87.84
## 13    13 Royal Dutch/Shell Group 133.50    8.40 100.72
## 10    10         Wal-Mart Stores 256.33    9.05 104.91
## 4      4               ExxonMobil 222.88   20.96 166.99
## 8      8            Toyota Motor 135.82    7.99 171.71
## 5      5                      BP 232.57   10.27 177.57
## 21    21         DaimlerChrysler 157.13    5.12 195.58
## 75    75              Ford Motor 164.20    0.76 312.56
## 29    29          General Motors 185.52    3.82 450.00
## 2      2        General Electric 134.19   15.59 626.93
```

```r
# plot
ggplot(selected_data, aes(x = assets, y = profits)) +
  geom_point(color = "blue") +
  ggtitle("Scatter plot") +
  theme(plot.title = element_text(hjust = 0.5), text = element_text(size = 14, family = "serif")) +
  xlab("Assets") +
  ylab("Profits")
```



(g)

- method 1(my function)

```
myKNN <- function(data, train_col, pred_col, k, scale, M){  #scale: scale() or min-max; M: mean or medi
  X <- data[, train_col]
  if(scale == "min-max"){
    # min-max normalization
    min <- apply(X, 2, min)
    max <- apply(X, 2, max)
    MIN <- matrix(rep(min, nrow(X)), ncol = length(min), byrow = T)
    MAX <- matrix(rep(max, nrow(X)), ncol = length(max), byrow = T)
    X <- (X - MIN) / (MAX - MIN)
  }else if(scale == "scale"){
    X <- scale(X)
  }
  # find index of missing value
  ind <- which(is.na(data[, pred_col]))
  # index of training samples
  train_id <- setdiff(1:nrow(X), ind)
  # operation w.r.t sample p
  for(p in ind) {
    dist <- replicate(nrow(X), -1)
    for(i in train_id) {
      temp <- 0
```

```r
      for(j in 1:length(train_col)) {
        temp <- temp + (X[i, train_col[j]] - X[p, train_col[j]])^2
      }
      dist[i] <- sqrt(temp)
    }
    # select k neighbors w.r.t sample p
    selid <- which(dist %in% sort(dist[dist >= 0])[1:k])
    # mean of k neighbors on pred_col    #or median or mode
    if(M == "mean"){
      data[p, pred_col] <- mean(data[selid, pred_col])
    }else if(M == "median"){
      data[p, pred_col] <- median(data[selid, pred_col])
    }
  }
  # show
  result <- data.frame(data[ind, -c(4, 5, 7, 8)])
  return(result)
}
```

```r
myKNN(F2000, c("sales", "assets", "marketvalue"), "profits", 10, "scale", "mean")
```

```
##      rank                  name        country profits
## 772   772                   AMP      Australia   0.449
## 1085 1085                   HHG United Kingdom   0.065
## 1091 1091                   NTL  United States   0.173
## 1425 1425      US Airways Group  United States  -0.060
## 1909 1909 Laidlaw International  United States  -0.022
```

```r
myKNN(F2000, c("sales", "assets", "marketvalue"), "profits", 10, "min-max", "mean")
```

```
##      rank                  name        country profits
## 772   772                   AMP      Australia   0.449
## 1085 1085                   HHG United Kingdom   0.065
## 1091 1091                   NTL  United States   0.173
## 1425 1425      US Airways Group  United States  -0.009
## 1909 1909 Laidlaw International  United States  -0.022
```

```r
myKNN(F2000, c("sales", "assets", "marketvalue"), "profits", 10, "scale", "median")
```

```
##      rank                  name        country profits
## 772   772                   AMP      Australia   0.510
## 1085 1085                   HHG United Kingdom   0.095
## 1091 1091                   NTL  United States   0.150
## 1425 1425      US Airways Group  United States   0.030
## 1909 1909 Laidlaw International  United States   0.055
```

```r
myKNN(F2000, c("sales", "assets", "marketvalue"), "profits", 10, "min-max", "median")
```

```
##      rank                  name        country profits
## 772   772                   AMP      Australia   0.510
```

```
## 1085 1085                   HHG United Kingdom   0.095
## 1091 1091                   NTL  United States   0.150
## 1425 1425      US Airways Group  United States   0.030
## 1909 1909 Laidlaw International  United States   0.055
```

- method 2(use package)

```
library(caret)
```

```
##      lattice
```

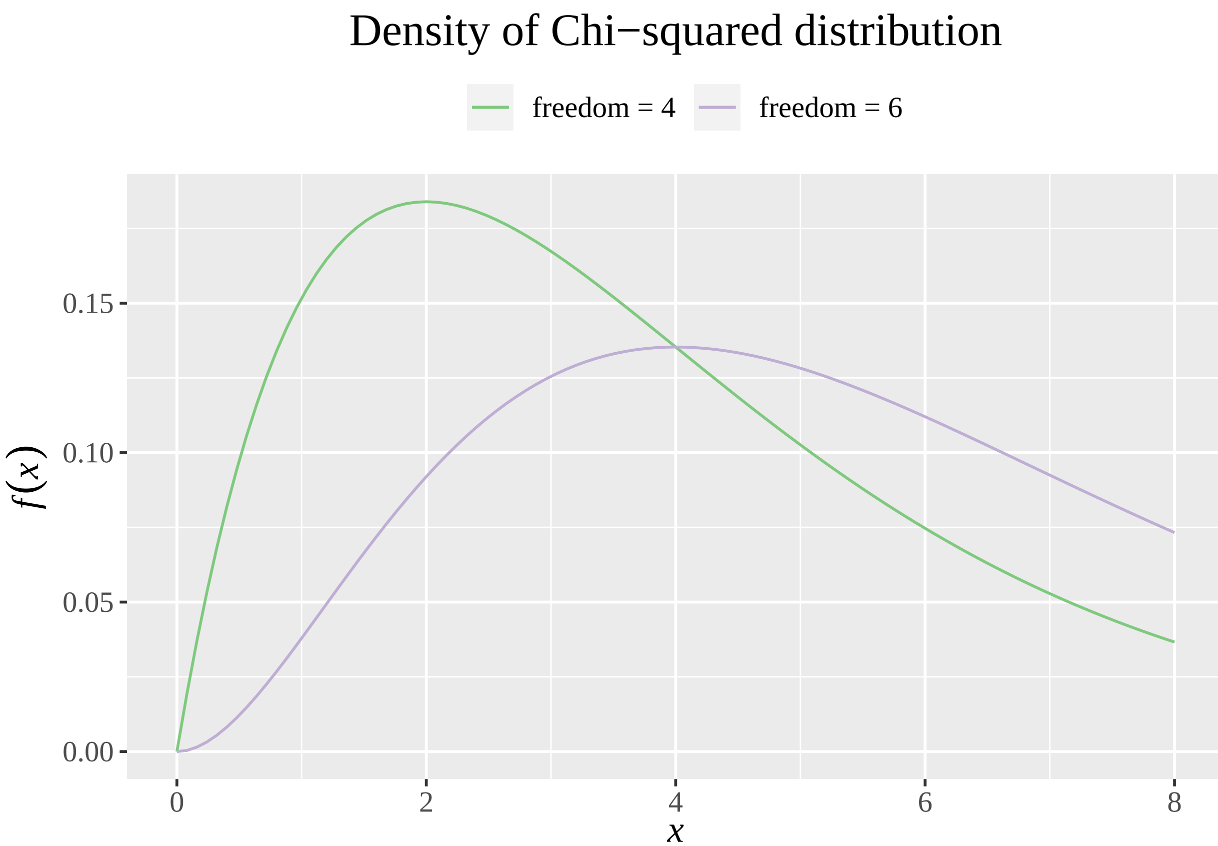
```
library(RANN)
library(lattice)
# save mean and sd for de-scaling
train_col <- c("sales", "assets", "marketvalue"); pred_col <- "profits"
ind <- which(is.na(F2000[, pred_col]))
mean <- mean(F2000[setdiff(1:nrow(F2000), ind), pred_col])
sd <- sd(F2000[setdiff(1:nrow(F2000), ind), pred_col])
# knn imputation
model <- preProcess(F2000[, c(train_col, pred_col)], method = "knnImpute", k = 10)
prediction <- predict(model, F2000)
# de-scaling
prediction[, pred_col] <- mean + sd * prediction[, pred_col]
# show
result <- data.frame(prediction[ind, -c(4, 5, 7, 8)])
result
```

```
##      rank                  name        country profits
## 772   772                   AMP      Australia   0.449
## 1085 1085                   HHG United Kingdom   0.065
## 1091 1091                   NTL  United States   0.173
## 1425 1425      US Airways Group  United States  -0.060
## 1909 1909 Laidlaw International  United States  -0.022
```

1.3

(a)

```
ggplot(data.frame(x = c(0, 8)), aes(x = x)) +
  ggtitle("Density of Chi-squared distribution") +
  stat_function(fun = dchisq, args = list(df = 4), aes(colour = "freedom = 4")) +
  stat_function(fun = dchisq, args = list(df = 6), aes(colour = "freedom = 6")) +
  scale_colour_brewer(palette = "Accent") +
  theme(plot.title = element_text(hjust = 0.5), legend.position = "top",
        text = element_text(size = 14, family = "serif")) +
  labs(colour = "") +
  xlab(expression(italic(x))) +
  ylab(expression(italic(f(x))))
```




```r
s1 = pchisq(7, df = 5) - pchisq(1, df = 5); s1
```

```
## [1] 0.7419255
```

```r
s2 = 1 - pchisq(3, df = 5); s2
```

```
## [1] 0.6999858
```

(b)

The cdf of X is given by

$$F(x) = \int_0^x \lambda e^{-\lambda t} dt = 1 - e^{-\lambda x}.$$

Then, $F(1) = 1 - e^{-\lambda}$, $F(6) = 1 - e^{-6\lambda}$.

```r
lambda <- -log(0.8)
F6 <- 1 - 0.8^6
sprintf('lambda:   %f', lambda)
```

```
## [1] "lambda:   0.223144"
```

```r
sprintf('Pr(X<=6): %f', F6)
```

```
## [1] "Pr(X<=6): 0.737856"
```

(c)

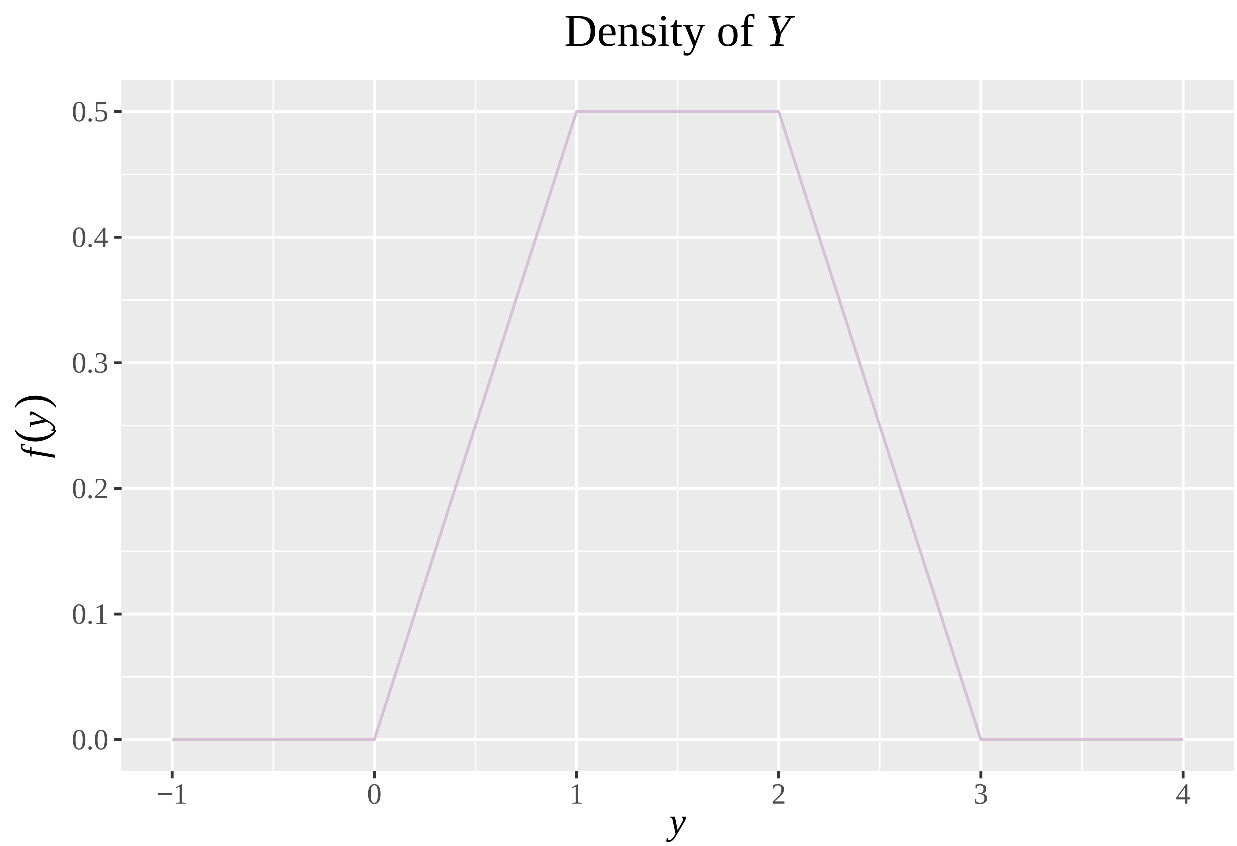
$X_i \sim U(0,1)$. Let $Y = X_1 + 2X_2, Z = X_1$, then$\frac{\partial(x_1,x_2)}{\partial(y,z)} =$

$$\begin{vmatrix} 0 & 1 \\ \frac{1}{2} & \frac{-1}{2} \end{vmatrix}$$

$= \frac{-1}{2}$. Then, $f_{Y,Z}(y,z) = \int_0^1 \int_0^1 f_{X_1,X_2}(x_1,x_2) \cdot \frac{1}{2} dx_1 dx_2 = \frac{1}{2}, \quad z \le y \le z+2,\ 0 \le z \le 1$ Hence, if $y < 0$, then $f_Y(y) = 0$; if $0 \le y < 1$, then$f_Y(y) = \int_0^y \frac{1}{2} dz = \frac{y}{2}$; if $1 \le y < 2$, then$f_Y(y) = \int_0^1 \frac{1}{2} dz = \frac{1}{2}$; if \$ 2 y <3,\$ then$f_Y(y) = \int_{y-2}^1 \frac{1}{2} dz = \frac{3-y}{2}$; if$y \ge 3$, then$f_Y(y) = 0$. Therefore, the pdf of Y is given by

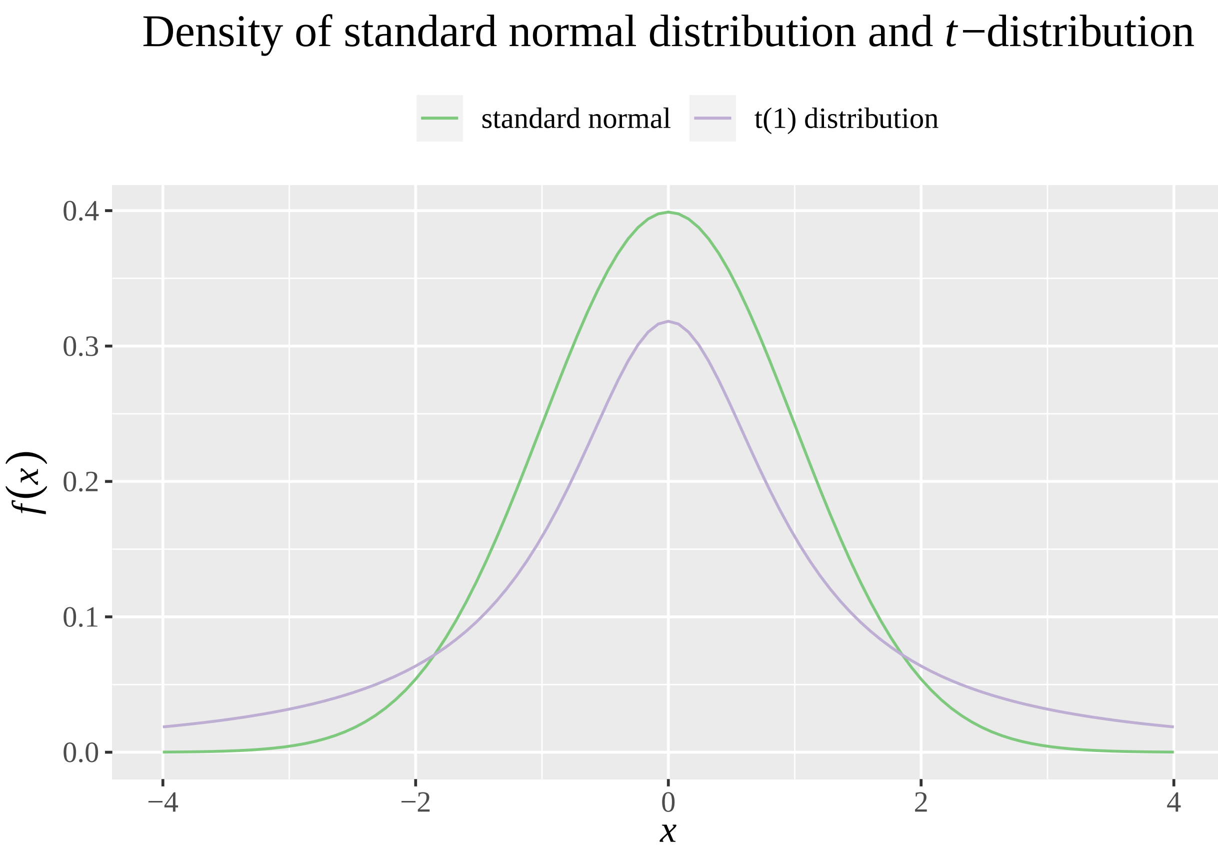
$$f_Y(y) = \begin{cases} 0 & y < 0, \\ \frac{y}{2} & 0 \le y < 1, \\ \frac{1}{2} & 1 \le y < 2, \\ \frac{3-y}{2} & 2 \le y < 3, \\ 0 & y \ge 3. \end{cases} \tag{1}$$

```r
x <- c(-1, 0, 1, 2, 3, 4)
y <- c(0, 0, 0.5, 0.5, 0, 0)
ggplot(data.frame(x, y), aes(x = x, y = y)) +
  geom_line(colour = 'thistle') +
  ggtitle(expression(paste("Density of ", italic(Y)))) +
  theme(plot.title = element_text(hjust = 0.5), text = element_text(size = 14, family = "serif")) +
  ylab(expression(italic(f(y)))) +
  xlab(expression(italic(y)))
```



(d)

```
ggplot(data.frame(x = c(-4, 4)), aes(x = x)) +
  ggtitle(expression(paste("Density of standard normal distribution and ", italic(t), "-distribution"))
  stat_function(fun = dnorm, args = list(0, 1), aes(colour = "standard normal")) +
  stat_function(fun = dt, args = list(1), aes(colour = "t(1) distribution")) +
  scale_colour_brewer(palette = "Accent") +
  theme(plot.title = element_text(hjust = 0.5), legend.position = "top",
        text = element_text(size = 14, family = "serif")) +
  labs(colour = "") +
  xlab(expression(italic(x))) +
  ylab(expression(italic(f(x))))
```



$t(1)$ distribution's tail is heavy.

(e)

Note that

$$1 = \sum_{x=0}^{\min\{m,n\}} \Pr(X = x) = \sum_{x=0}^{\min\{m,n\}} \frac{\binom{m}{x}\binom{N-m}{n-x}}{\binom{N}{n}}.$$

That is,

$$\sum_{x=0}^{\min\{m,n\}} \binom{m}{x}\binom{N-m}{n-x} = \binom{N}{n}.$$

The expectation of X is given by

$$E(X) = \sum_{x=0}^{\min\{m,n\}} x\Pr(X = x) = \frac{1}{\binom{N}{n}} \sum_{x=1}^{\min\{m,n\}} x\binom{m}{x}\binom{N-m}{n-x}.$$

Using $x\binom{m}{x} = m\binom{m-1}{x-1}$, we have

$$\begin{aligned} E(X) &= \frac{1}{\binom{N}{n}} \sum_{t=0}^{\min\{m-1,n-1\}} m\binom{m-1}{t}\binom{(N-1)-(m-1)}{(n-1)-t} \quad (t = x-1) \\ &= \frac{m}{\binom{N}{n}} \sum_{t=0}^{\min\{m-1,n-1\}} \binom{m-1}{t}\binom{(N-1)-(m-1)}{(n-1)-t} \\ &= \frac{m}{\binom{N}{n}} \binom{N-1}{n-1} \\ &= \frac{mn}{N}. \end{aligned}$$

Similarly, using $x^2 = x(x-1)+x$ and the same skill, we can obtain that $E(X^2) = \frac{mn}{N} + \frac{mn(m-1)(n-1)}{N(N-1)}$. Then

$$\mathrm{Var}(X) = E(X^2) - (EX)^2 = \frac{mn(N-m)(N-n)}{N^2(N-1)}.$$

```r
p <- replicate(9, 0)
name <- p; E <- 0; E2 <- 0
for(k in 0:8){
  name[k+1] <- paste("Pr(X = ", k, ")", sep = "")
  p[k+1] <- (choose(28, k) * choose(17, 8-k)) / choose(45, 8)
  E <- E + k * p[k+1]
  E2 <- E2 + k^2 * p[k+1]
}
table <- data.frame(name, p); table
```

```
##         name            p
## 1 Pr(X = 0) 0.0001127796
## 2 Pr(X = 1) 0.0025262627
## 3 Pr(X = 2) 0.0217028933
## 4 Pr(X = 3) 0.0940458711
## 5 Pr(X = 4) 0.2260718056
## 6 Pr(X = 5) 0.3100413334
## 7 Pr(X = 6) 0.2376983556
## 8 Pr(X = 7) 0.0933814969
## 9 Pr(X = 8) 0.0144192017
```

```r
Var <- E2 - E^2
paste("Expectation:", E)
```

```
## [1] "Expectation: 4.97777777777778"
```

```r
paste("Variation:  ", Var)
```

```
## [1] "Variation:   1.58132435465768"
```

(f)

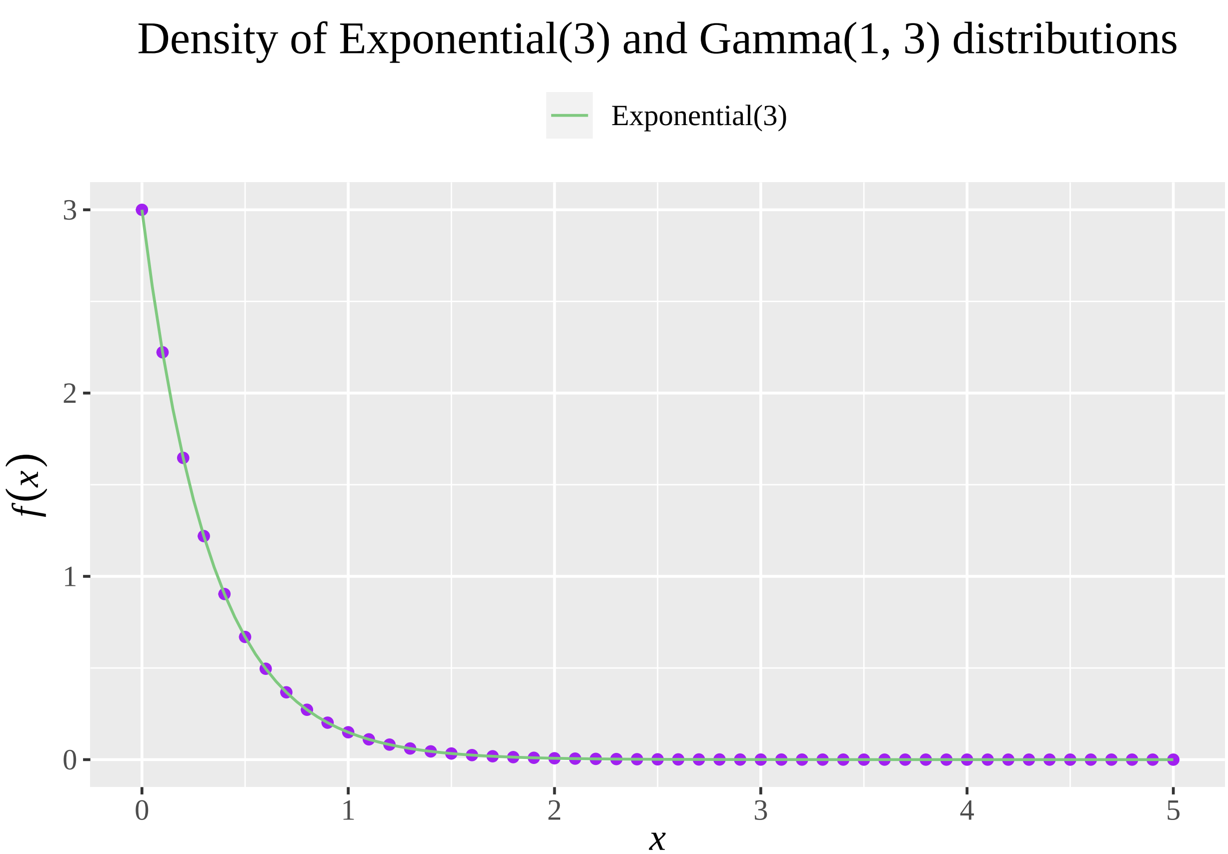
$\Pr(X > 8) \le 0.4$ is equivalent to $\sum_{k=0}^{8} \Pr(X = k) \ge 0.6$.

```r
f <- function(lambda){sum(dpois(0:8, lambda)) - 0.6}
if(f(1) > 0 && f(10) < 0){
  lambda <- uniroot(f, lower = 1, upper = 10)[[1]]; lambda
}
```

```
## [1] 7.946606
```

(g)

```r
x <- seq(0, 5, 0.1)
y <- dgamma(x, 1, 3)
ggplot(data.frame(x, y), aes(x = x, y = y)) +
  geom_point(color = 'purple') +
  stat_function(fun = dexp, args = list(3), aes(colour = "Exponential(3)")) +
  ggtitle("Density of Exponential(3) and Gamma(1, 3) distributions") +
  scale_colour_brewer(palette="Accent") +
  theme(plot.title = element_text(hjust = 0.5), legend.position = "top",
        text = element_text(size = 14, family = "serif")) +
  labs(colour = "") +
  xlab(expression(italic(x))) +
  ylab(expression(italic(f(x))))
```



```
# Purple points are generated from Gamma(1, 3) distribution
```

(h)

```
prob <- pretty(0:1, 20); prob[21] <- 1  # 1 - prob[21] = -2.220446e-16 without this assignment
size <- replicate(21, 0)
for(i in 1:21){
  n <- 1; p <- 1
  while(p > 1 - prob[i]){
    n <- n + 1
    p <- p * (365 - n + 1) / 365
  }
  size[i] <- n
}
result <- data.frame(size, prob)
result
```

```
##    size prob
## 1     1 0.00
## 2     7 0.05
## 3    10 0.10
## 4    12 0.15
## 5    14 0.20
## 6    15 0.25
## 7    17 0.30
## 8    19 0.35
## 9    20 0.40
## 10   22 0.45
## 11   23 0.50
## 12   25 0.55
## 13   27 0.60
## 14   28 0.65
## 15   30 0.70
## 16   32 0.75
## 17   35 0.80
## 18   38 0.85
## 19   41 0.90
## 20   47 0.95
## 21  366 1.00
```

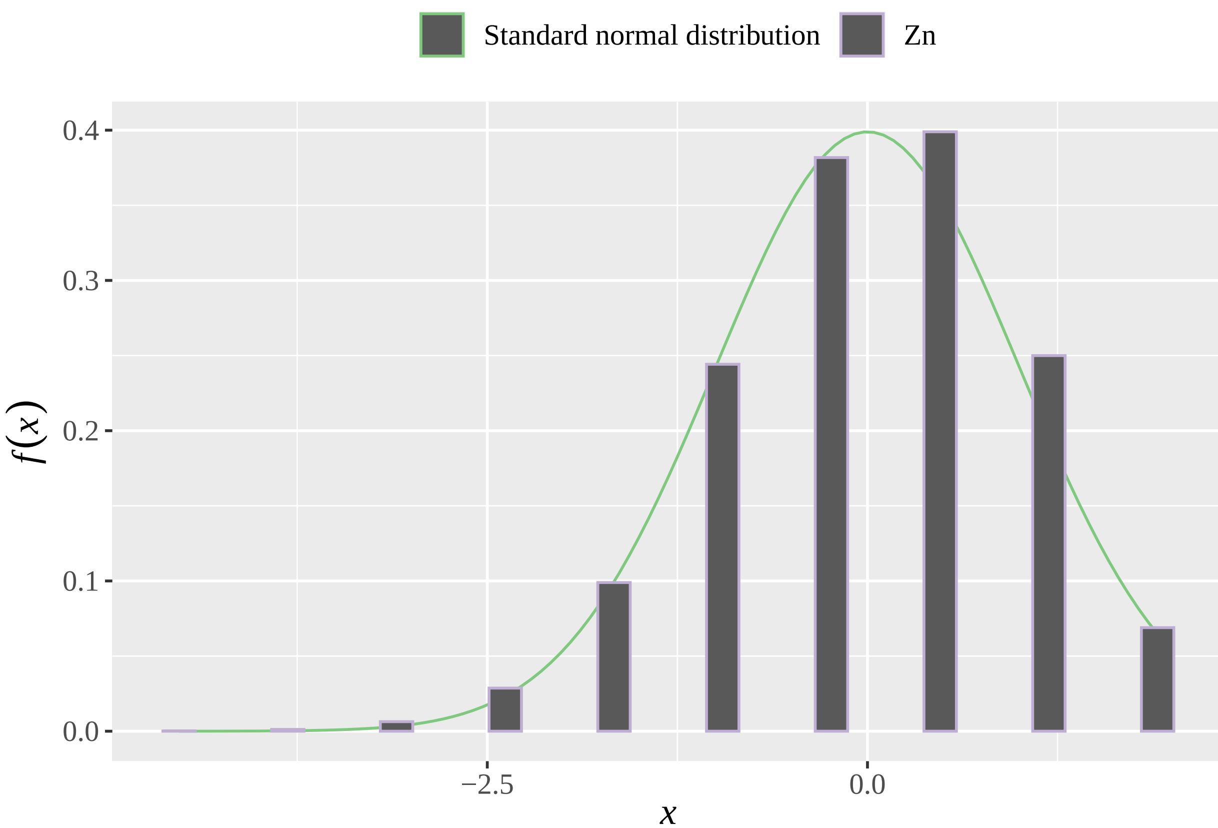
(i)

- method 1 (visualization)

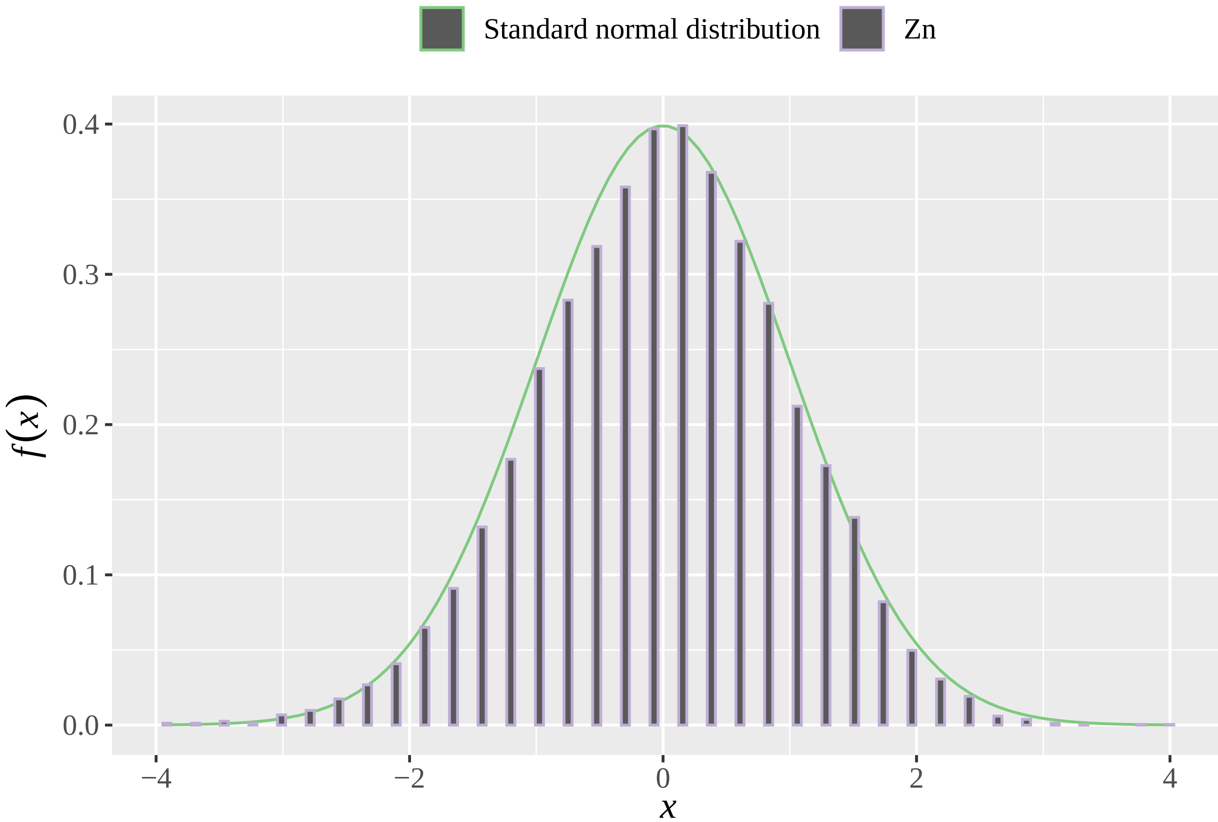
```
myplot <- function(n, prob){
  times <- 10000
  x <- rbinom(times, n, prob)
  z <- (x - n * prob) / (sqrt(n * prob * (1 - prob)))
  p <- ggplot(data.frame(value = z), aes(x = value)) +
  stat_function(fun = dnorm, aes(colour = 'Standard normal distribution')) +
```

```r
  geom_bar(aes(y =..count../ (sqrt(2*pi)* max(..count..)) , colour = 'Zn')) +
  scale_colour_brewer(palette = "Accent") +
  theme(plot.title = element_text(hjust = 0.5), legend.position = "top",
        text = element_text(size = 14, family = "serif")) +
  labs(colour = "") +
  xlab(expression(italic(x))) +
  ylab(expression(italic(f(x))))
return(p)
}
```

```r
prob <- runif(1, min = 0, max = 1)
myplot(10, prob)
```




```r
myplot(100, prob)
```



- method 2 (calculate) - a and b not fixed

```
# simulation times of Zn
times <- 5000
# randomly choose p
p <- runif(1, min = 0, max = 1)
# calculate Pr(a < Zn <= b) for n = 10, 100, 1000, 10000
simulation <- rep(0, 4); true <- rep(0, 4); a<- rep(0, 4); b <- rep(0, 4)
for(n in c(10, 100, 1000, 10000)){
  x <- rbinom(times, n, p)
  z <- (x - n * p) / (sqrt(n * p * (1 - p)))
  # randomly choose a and b among the range of z
  nums <- runif(2, min = min(z), max = max(z))
  a[log10(n)] <- min(nums); b[log10(n)] <- max(nums)
  simulation[log10(n)] <- length(z[z > a[log10(n)] & z <= b[log10(n)]]) / length(z)
  # value of integral
  true[log10(n)] <- pnorm(b[log10(n)]) - pnorm(a[log10(n)])
}
# show
n <- c(10, 100, 1000, 10000)
comparision <- data.frame(n, a, b, simulation, true); comparision
```

```
##       n          a         b simulation       true
## 1    10 -3.6963408 0.5428541     0.6038 0.70627551
## 2   100 -0.9588259 2.5895012     0.8288 0.82637102
```

```
## 3  1000 -0.2954270 -0.1637378     0.0568 0.05113494
## 4 10000 -2.7829912  0.1807639     0.5816 0.56903053
```

– a and b fixed

```r
a <- -1; b <- 1
times <- 5000
p <- runif(1, min = 0, max = 1)
simulation <- rep(0, 4); true <- rep(0, 4)
for(n in c(10, 100, 1000, 10000)){
  x <- rbinom(times, n, p)
  z <- (x - n * p) / (sqrt(n * p * (1 - p)))
  simulation[log10(n)] <- length(z[z > a & z <= b]) / length(z)
  true[log10(n)] <- pnorm(b) - pnorm(a)
}
# show
n <- c(10, 100, 1000, 10000)
comparision <- data.frame(n, a, b, simulation, true)
comparision
```

```
##       n  a b simulation      true
## 1    10 -1 1     0.6690 0.6826895
## 2   100 -1 1     0.6940 0.6826895
## 3  1000 -1 1     0.6898 0.6826895
## 4 10000 -1 1     0.6832 0.6826895
```

(j)

- method 1

```r
# lambda: lambda; times: times of the generation of Sn for each n; K: k = 1,...,K
poissonlimit <- function(lambda, times, K){
  simulation <- matrix(rep(0, 5*K), 5, K)

  for(n in c(10, 100, 1000, 10000)){
    sn <- rbinom(times, n, lambda / n)
    for(k in 1:K){
      simulation[log10(n), k] <- length(sn[sn == k]) / length(sn)
    }
  }
  # show
  result <- data.frame(c(10, 100, 1000, 10000, "true"), simulation)
  colnames(result) <- c("n\\k", 1:9)
  result[5, 2:10] <- dpois(1:9, lambda)
  return(result)
}
```

```r
poissonlimit(3, 5000, 9)
```

```
##    n\\k         1         2         3         4         5          6          7
## 1    10 0.1276000 0.2270000 0.2746000 0.1974000 0.0966000 0.03380000 0.00840000
```

```
## 2   100 0.1454000 0.2234000 0.2316000 0.1750000 0.0970000 0.04680000 0.02060000
## 3  1000 0.1454000 0.2320000 0.2126000 0.1706000 0.1018000 0.04720000 0.02280000
## 4 10000 0.1522000 0.2208000 0.2268000 0.1612000 0.1002000 0.05060000 0.02120000
## 5  true 0.1493612 0.2240418 0.2240418 0.1680314 0.1008188 0.05040941 0.02160403
##             8           9
## 1 0.001200000 0.000200000
## 2 0.006600000 0.002200000
## 3 0.010600000 0.001400000
## 4 0.009000000 0.004000000
## 5 0.008101512 0.002700504
```

```
poissonlimit(5, 5000, 9)
```

```
##    n\\k          1          2         3         4         5         6         7
## 1    10 0.00900000 0.04740000 0.1178000 0.2158000 0.2370000 0.2008000 0.1162000
## 2   100 0.02600000 0.08320000 0.1288000 0.1840000 0.1760000 0.1534000 0.1094000
## 3  1000 0.03380000 0.08240000 0.1406000 0.1602000 0.1832000 0.1460000 0.1086000
## 4 10000 0.03180000 0.08360000 0.1400000 0.1840000 0.1756000 0.1466000 0.1032000
## 5  true 0.03368973 0.08422434 0.1403739 0.1754674 0.1754674 0.1462228 0.1044449
##            8          9
## 1 0.04340000 0.00960000
## 2 0.06680000 0.03660000
## 3 0.06120000 0.03920000
## 4 0.06520000 0.03320000
## 5 0.06527804 0.03626558
```

```
poissonlimit(7, 5000, 9)
```

```
##    n\\k           1          2          3          4         5         6
## 1    10 0.000200000 0.00060000 0.00860000 0.03480000 0.0988000 0.2008000
## 2   100 0.004800000 0.02000000 0.04660000 0.09340000 0.1312000 0.1532000
## 3  1000 0.006000000 0.02020000 0.05340000 0.09520000 0.1204000 0.1496000
## 4 10000 0.008600000 0.02040000 0.04660000 0.08860000 0.1270000 0.1542000
## 5  true 0.006383174 0.02234111 0.05212925 0.09122619 0.1277167 0.1490028
##           7         8         9
## 1 0.2670000 0.2356000 0.1240000
## 2 0.1488000 0.1312000 0.0988000
## 3 0.1490000 0.1306000 0.1040000
## 4 0.1510000 0.1276000 0.1088000
## 5 0.1490028 0.1303774 0.1014047
```

```
poissonlimit(9, 5000, 9)
```

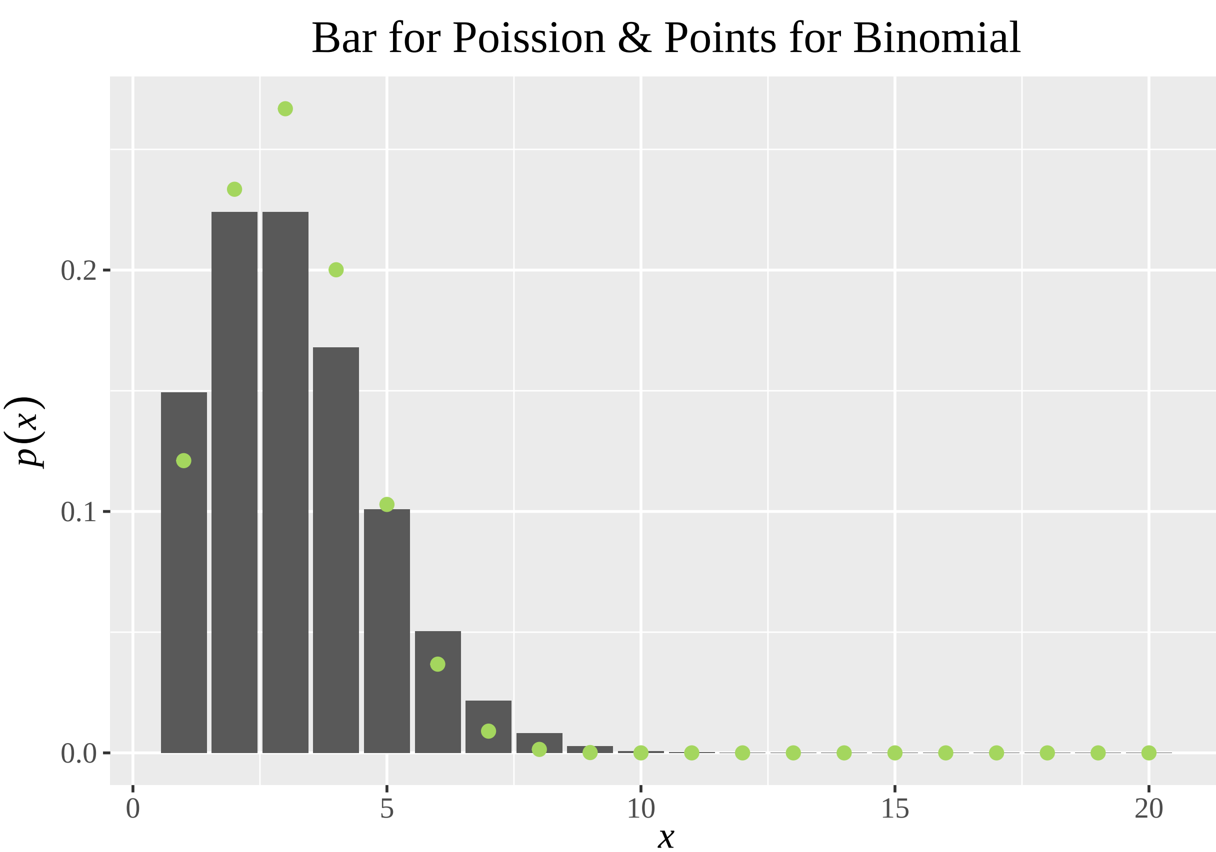
```
##    n\\k           1           2          3          4          5          6
## 1    10 0.000000000 0.000000000 0.00000000 0.00000000 0.00200000 0.00760000
## 2   100 0.001200000 0.002600000 0.01360000 0.03100000 0.05700000 0.08340000
## 3  1000 0.001800000 0.004000000 0.01720000 0.03180000 0.06220000 0.08780000
## 4 10000 0.001000000 0.004800000 0.01520000 0.03100000 0.05960000 0.09400000
## 5  true 0.001110688 0.004998097 0.01499429 0.03373716 0.06072688 0.09109032
##           7         8         9
## 1 0.0512000 0.1924000 0.3974000
## 2 0.1148000 0.1344000 0.1418000
```

```
## 3 0.1082000 0.1382000 0.1406000
## 4 0.1160000 0.1278000 0.1348000
## 5 0.1171161 0.1317556 0.1317556
```

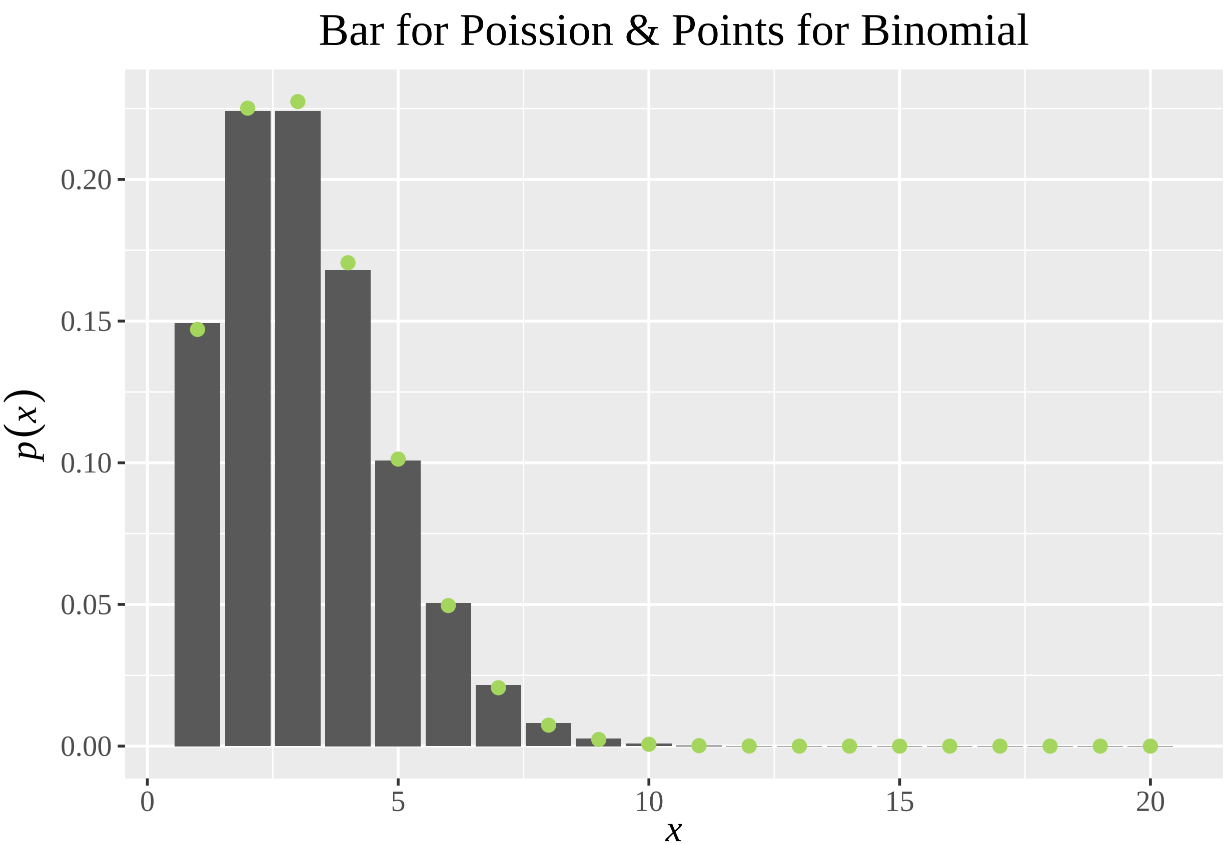
- method 2 (visualization)

```
pois_binom <- function(lambda, n){
  p <- lambda / n
  x <- 1:20
  y_pois <- dpois(x, lambda)
  y_binom <- dbinom(x, n, p)
  pp <- ggplot(data.frame(x, y_pois, y_binom)) +
    geom_bar(aes(x = x, y = y_pois), stat = 'identity') +
    geom_point(aes(x = x, y = y_binom), col = '#A4D65E', size = 2) +
    ggtitle("Bar for Poission & Points for Binomial") +
    theme(plot.title = element_text(hjust = 0.5), legend.position = "top",
        text = element_text(size = 14, family = "serif")) +
    xlab(expression(italic(x))) +
    ylab(expression(italic(p(x))))
  return(pp)
}
```

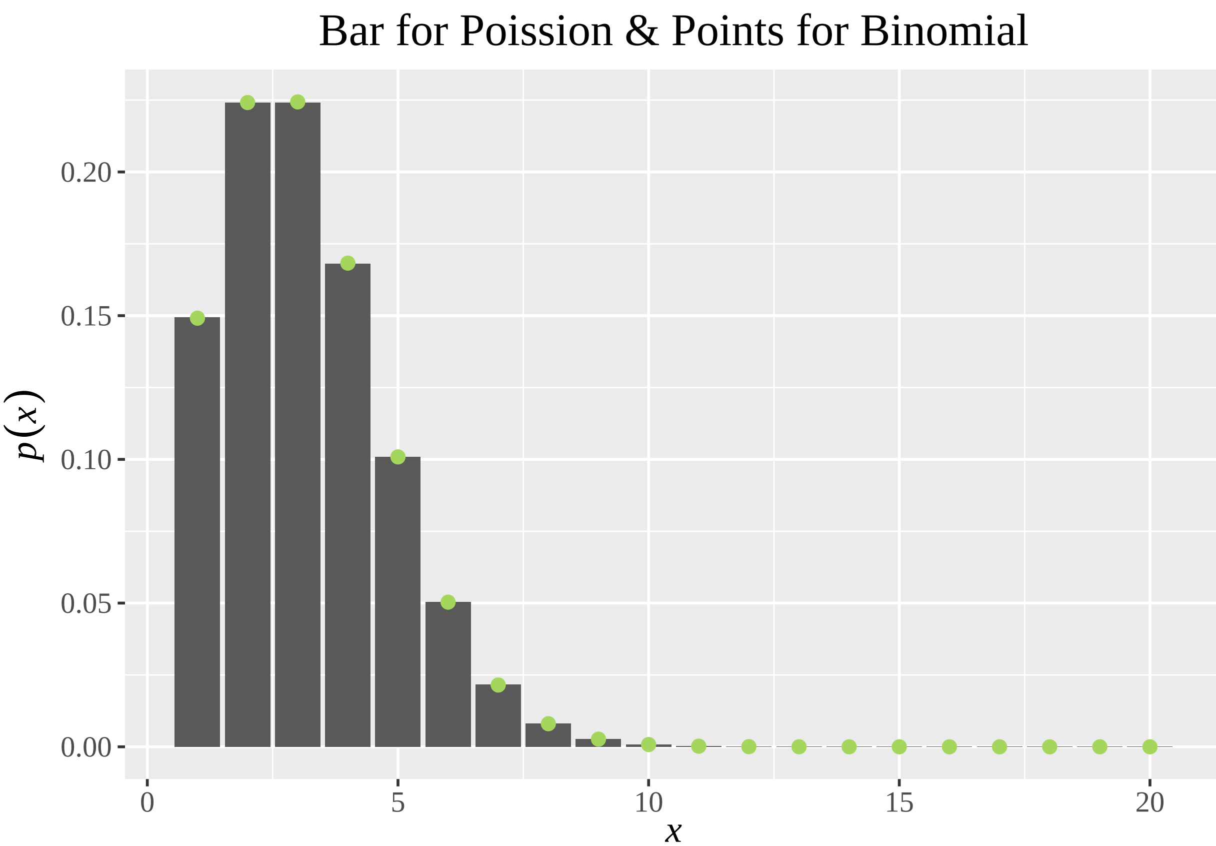
```
pois_binom(3, 10)
```



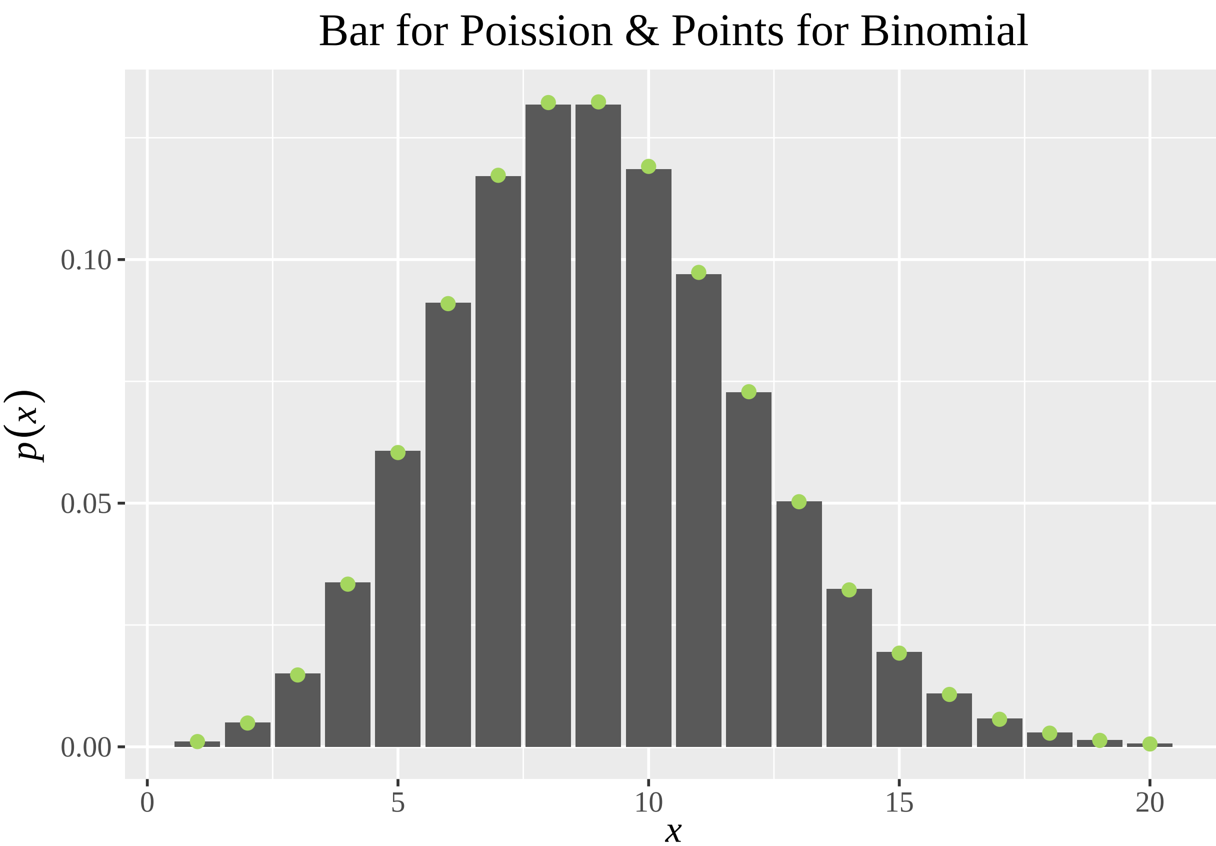
```r
pois_binom(3, 100)
```




```r
pois_binom(3, 1000)
```




```
##
pois_binom(9, 1000)
```



(k)

Note that in R, the geometric distribution with $prob = p$ has density

$$p(x) = p(1-p)^x,$$

for $x = 0, 1, 2, 3, ...$, which is different from what we tipically learned in class:

$$p(x) = p(1-p)^{x-1},$$

for $x = 1, 2, 3, ...$. Therefore, we should use $pgeom(m+n+1)$ instead of $pgeom(m+n)$ for geometric distribution.

- method 1 (cdf method)

```
# p: p or lambda; type: geometric or poisson
memoryless_1 <- function(m, n, p, type){
  if(type == 'Geometric'){
    if(round((1 - pgeom(m + n + 1, p)), 4) == round(((1 - pgeom(m, p)) * (1 - pgeom(n, p))), 4)){
      print('Geometric  distribution has memoryless  property')
    }else{
       print('Geometric  distribution does not have memoryless  property')
    }
  }else if(type == 'Poisson'){
    if(round((1 - ppois(m + n, p)), 4) == round(((1 - ppois(m, p)) * (1 - ppois(n, p))), 4)){
```

```
      print('Poisson  distribution has memoryless  property')
    }else{
       print('Poisson  distribution does not have memoryless  property')
    }
  }
}
```

```
memoryless_1(5, 7, 0.2, 'Geometric')
```

```
## [1] "Geometric  distribution has memoryless  property"
```

```
memoryless_1(11, 103, 0.4, 'Geometric')
```

```
## [1] "Geometric  distribution has memoryless  property"
```

```
memoryless_1(75, 49, 0.6, 'Geometric')
```

```
## [1] "Geometric  distribution has memoryless  property"
```

```
memoryless_1(5, 7, 3, 'Poisson')
```

```
## [1] "Poisson  distribution does not have memoryless  property"
```

```
memoryless_1(11, 13, 7, 'Poisson')
```

```
## [1] "Poisson  distribution does not have memoryless  property"
```

```
memoryless_1(9, 40, 21, 'Poisson')
```

```
## [1] "Poisson  distribution does not have memoryless  property"
```

- method 2 (simulation method)

```
# p: p or lambda; T: simulation times; k: number of x per time; type: geometric or poisson
memoryless_2 <- function(p, T, k, type){
  result <- matrix(rep(0, 4*T), T, 4)
  for(t in 1:T){
    if(type == "geometric"){
      x <- rgeom(k, p)
    }else if(type == "Poisson"){
      x <- rpois(k, p)
    }
    # in order to get a good simulation, m and n are selected as follows
    m <- round(runif(1, min = mean(x) - sd(x), max = mean(x) + sd(x)))
    n <- round(runif(1, min = mean(x) - sd(x), max = mean(x) + sd(x)))
    if(type == "geometric"){
      p_condition <- length(x[x > m + n + 1]) / length(x[x > m])
    }else if(type == "Poisson"){
```

```r
      p_condition <- length(x[x > m + n]) / length(x[x > m])
    }
    p2 <- length(x[x > n]) / length(x)
    result[t, ] <- c(p_condition, p2, m, n)
  }
  result <- data.frame(result)
  colnames(result) <- c("Pr(X > m+n|X > m)", "Pr(X > n)", "m", "n")
  return(result)
}
```

```r
memoryless_2(0.5, 10, 10000, "geometric")
```

```
##    Pr(X > m+n|X > m) Pr(X > n) m n
## 1          0.5031822    0.5040 1 0
## 2          0.1203427    0.1239 0 2
## 3          0.1239804    0.1226 1 2
## 4          0.2543434    0.2530 0 1
## 5          0.2457364    0.2591 2 1
## 6          0.2444176    0.2449 0 1
## 7          0.2478805    0.2492 0 1
## 8          0.1338170    0.1263 1 2
## 9          0.2532154    0.2488 1 1
## 10         0.2544868    0.2488 0 1
```

```r
memoryless_2(0.1, 10, 10000, "geometric")
```

```
##    Pr(X > m+n|X > m) Pr(X > n)  m  n
## 1          0.3997904    0.3817  8  8
## 2          0.2262166    0.2281 13 13
## 3          0.2036842    0.1949  8 14
## 4          0.2067757    0.2038  9 14
## 5          0.1935720    0.1849 18 15
## 6          0.4340220    0.4326  2  7
## 7          0.5439513    0.5318 13  5
## 8          0.4726270    0.4794 12  6
## 9          0.1744361    0.1647 18 16
## 10         0.1853324    0.1808 11 15
```

```r
memoryless_2(0.01, 10, 10000, "geometric")
```

```
##    Pr(X > m+n|X > m) Pr(X > n)   m   n
## 1          0.1622230    0.1649  11 181
## 2          0.3026886    0.3000  76 120
## 3          0.9893417    0.9898  73   0
## 4          0.3948508    0.3924  12  91
## 5          0.4243964    0.4240 141  84
## 6          0.2287380    0.2296 123 148
## 7          0.7437568    0.7499 168  27
## 8          0.4268005    0.4103 174  87
## 9          0.5213155    0.5013 177  67
## 10         0.2443688    0.2309  94 146
```

From the result, we can see that geometric distribution has memoryless property.

```
memoryless_2(1, 10, 10000, "Poisson")
```

```
##    Pr(X > m+n|X > m) Pr(X > n) m n
## 1         0.23871734    0.2743 2 1
## 2         0.30715397    0.2572 1 1
## 3         1.00000000    0.6270 1 0
## 4         1.00000000    0.6380 1 0
## 5         0.23498695    0.2711 2 1
## 6         0.29202871    0.2647 1 1
## 7         0.29494278    0.2709 1 1
## 8         0.04516129    0.0775 2 2
## 9         0.30152091    0.2630 1 1
## 10        1.00000000    0.6345 0 0
```

```
memoryless_2(5, 10, 10000, "Poisson")
```

```
##    Pr(X > m+n|X > m) Pr(X > n) m n
## 1        0.041185266    0.2414 3 6
## 2        0.098089690    0.3882 3 5
## 3        0.178847458    0.5576 3 4
## 4        0.017159199    0.1277 3 7
## 5        0.002136752    0.1319 6 7
## 6        0.111690246    0.7284 7 3
## 7        0.011252009    0.1345 4 7
## 8        0.017736046    0.2377 5 6
## 9        0.024827836    0.2400 4 6
## 10       0.008957655    0.2456 6 6
```

```
memoryless_2(10, 10, 10000, "Poisson")
```

```
##    Pr(X > m+n|X > m) Pr(X > n)  m  n
## 1       0.0101241019    0.6620 11  8
## 2       0.0014295926    0.3051 10 11
## 3       0.0426710585    0.6679  8  8
## 4       0.0000000000    0.2066 12 12
## 5       0.0106737825    0.6671 11  8
## 6       0.0030216045    0.2090  8 12
## 7       0.0025483436    0.2099  8 12
## 8       0.0000000000    0.1366 11 13
## 9       0.0006680027    0.2045 11 12
## 10      0.0033670034    0.4127 11 10
```

```
memoryless_2(50, 10, 10000, "Poisson")
```

```
##    Pr(X > m+n|X > m) Pr(X > n)  m  n
## 1                  0    0.2646 53 54
## 2                  0    0.6812 49 46
## 3                  0    0.7333 53 45
## 4                  0    0.1763 50 56
```

```
## 5                  0    0.7333 55 45
## 6                  0    0.7777 54 44
## 7                  0    0.5178 55 49
## 8                  0    0.7821 56 44
## 9                  0    0.7350 48 45
## 10                 0    0.7856 45 44
```

From the result, Poisson distribution does not have memoryless property.