Statistical Calculation and Software

Assignment 1

Hanbin Liu 11912410

1.1

(a)

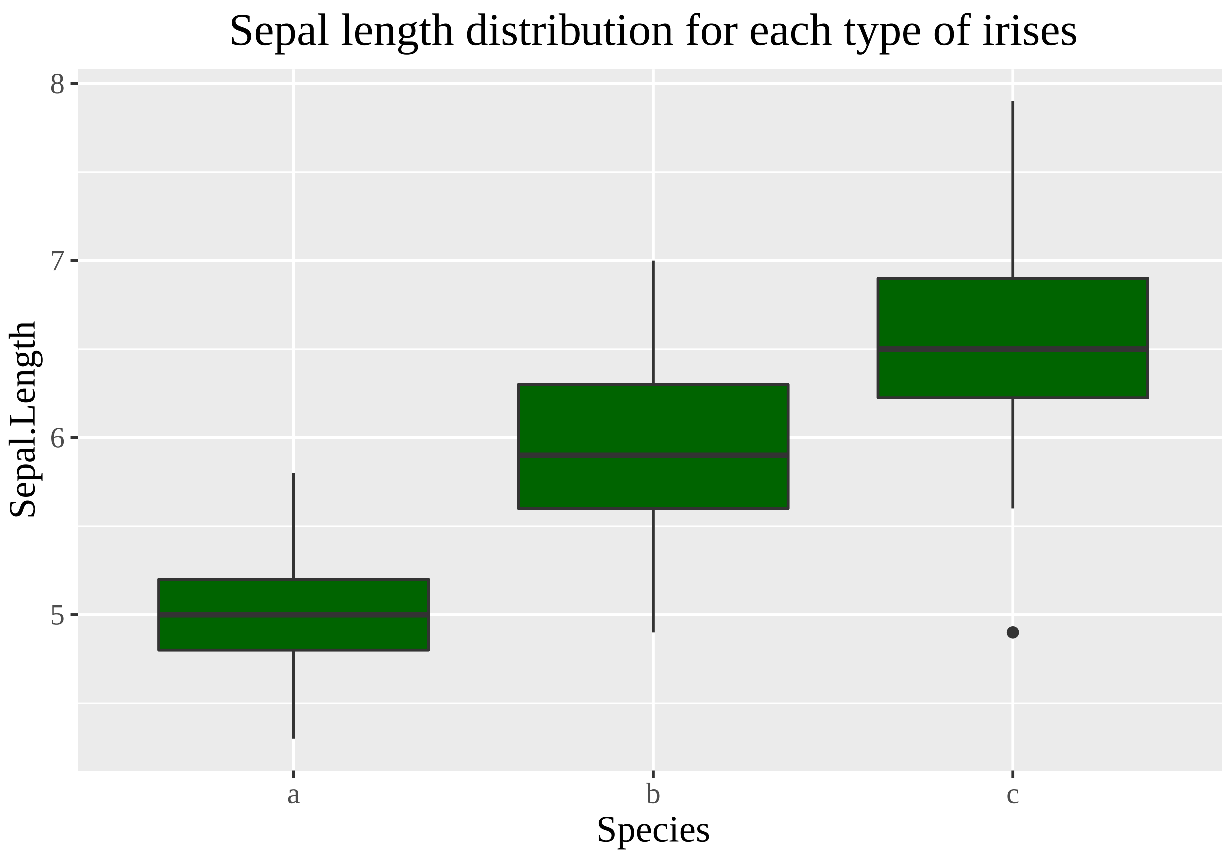
```r
data('iris')
```

```r
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)

```r
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"))
```

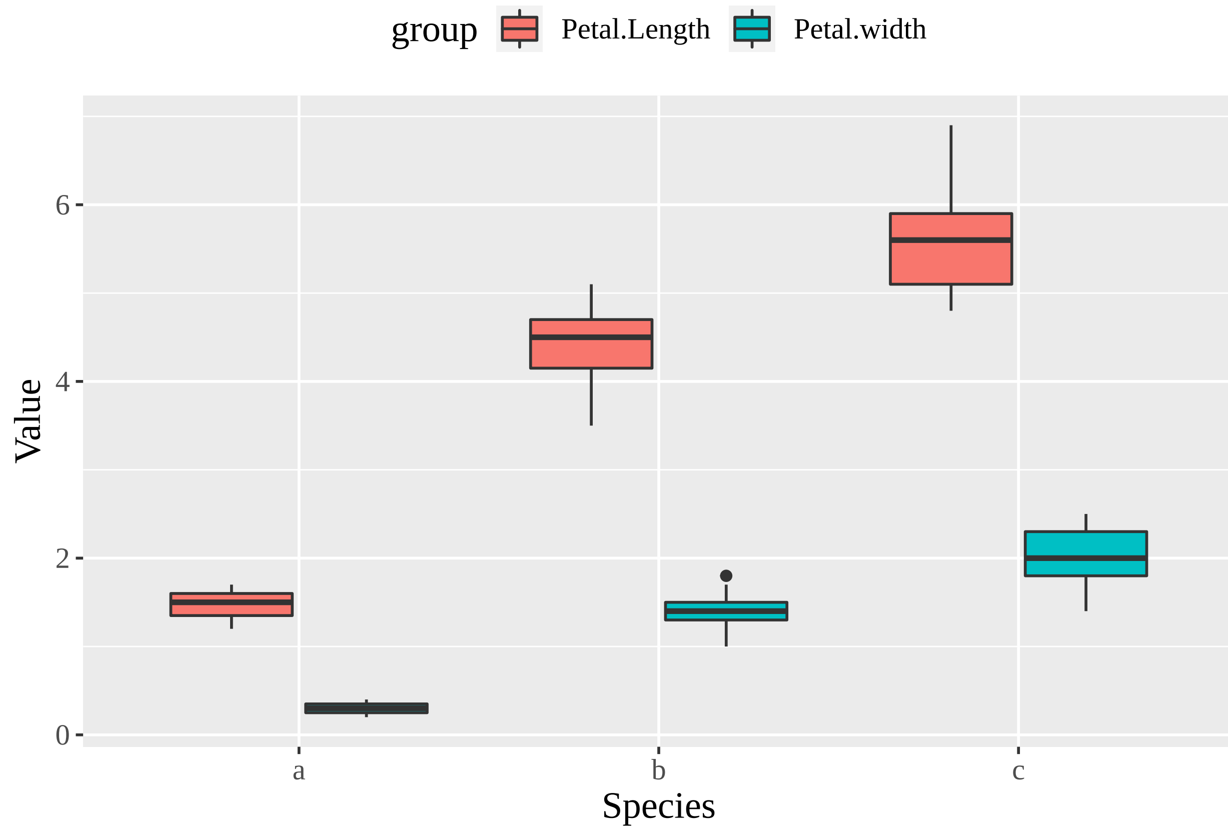


(c)

```
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
```

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

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

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

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

(c)

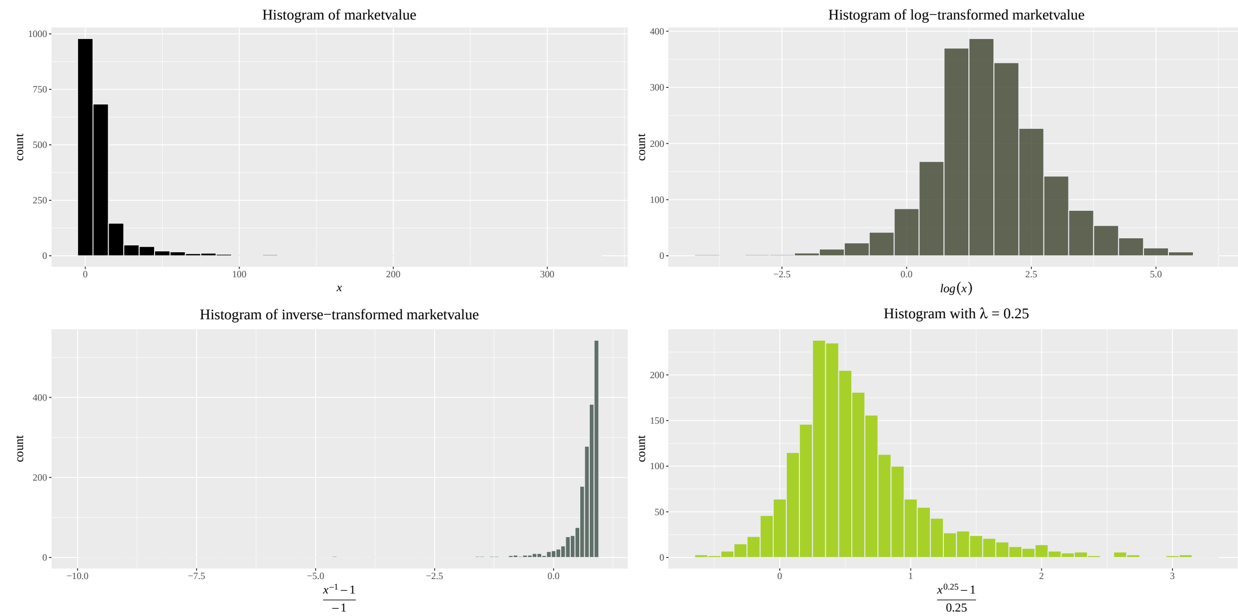
```
library(patchwork)
p1 <- ggplot(F2000, aes(x = marketvalue)) +
  geom_histogram(binwidth = 10,
                 fill = "#000000",
                 color = "#e9ecef") +
  ggtitle("Histogram of marketvalue") +
  theme(plot.title = element_text(hjust = 0.5),
        text = element_text(size = 14, family = "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)
}
```

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

(f)

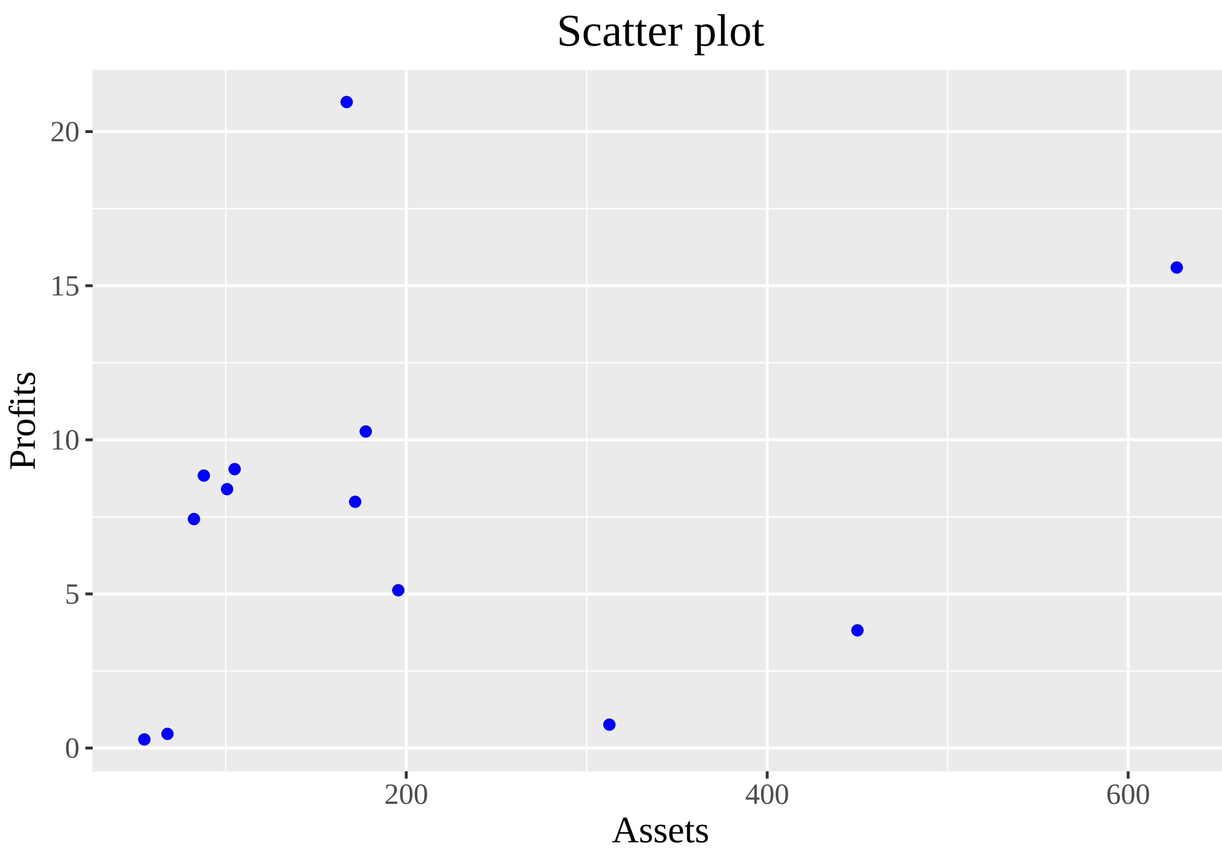
```r
# 2 ways
way_1 <- F2000[F2000$sales > 100,]
way_2 <- F2000[which(F2000$sales > 100),]
# sort
selected_data <- way_1[-c(3, 4, 8)]
ind_1 <- order(selected_data$sales, decreasing = TRUE)
selected_data[ind_1,]
```

```
##     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)

```r
myKNN <-
  function(data, train_col, pred_col, k, scale, M) {
    #scale: scale() or min-max; M: mean or median
    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
        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)
  }
```

```
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
```

```
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
```

```
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
```

```
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 <-
```

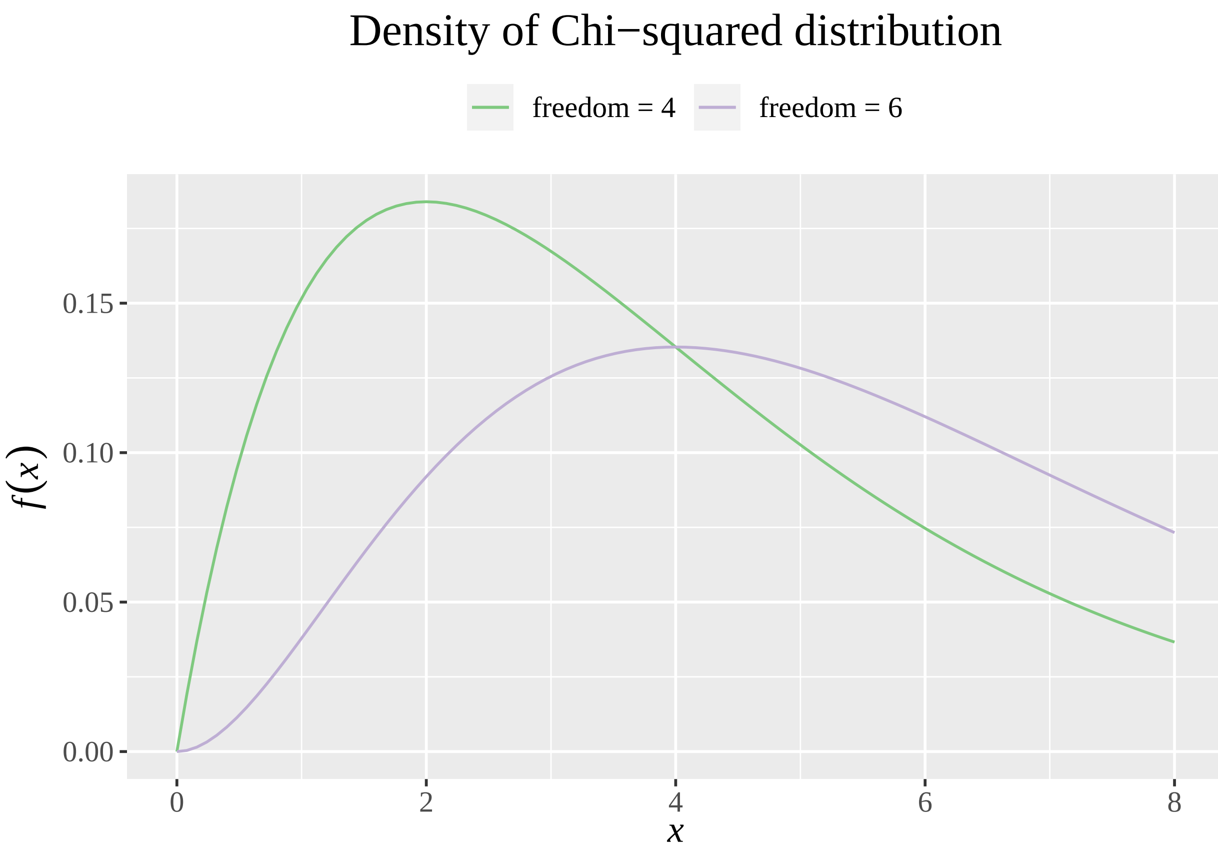
```r
  preProcess(F2000[, c(train_col, pred_col)], method = "knnImpute", k = 10)
prediction <- predict(model, F2000)
# de-scaling
prediction[, pred_col] <- mean + sd * prediction[, pred_col]
# show
result <- data.frame(prediction[ind,-c(4, 5, 7, 8)])
result
```

```
##      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)

```r
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))))
```




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

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

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

```
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)

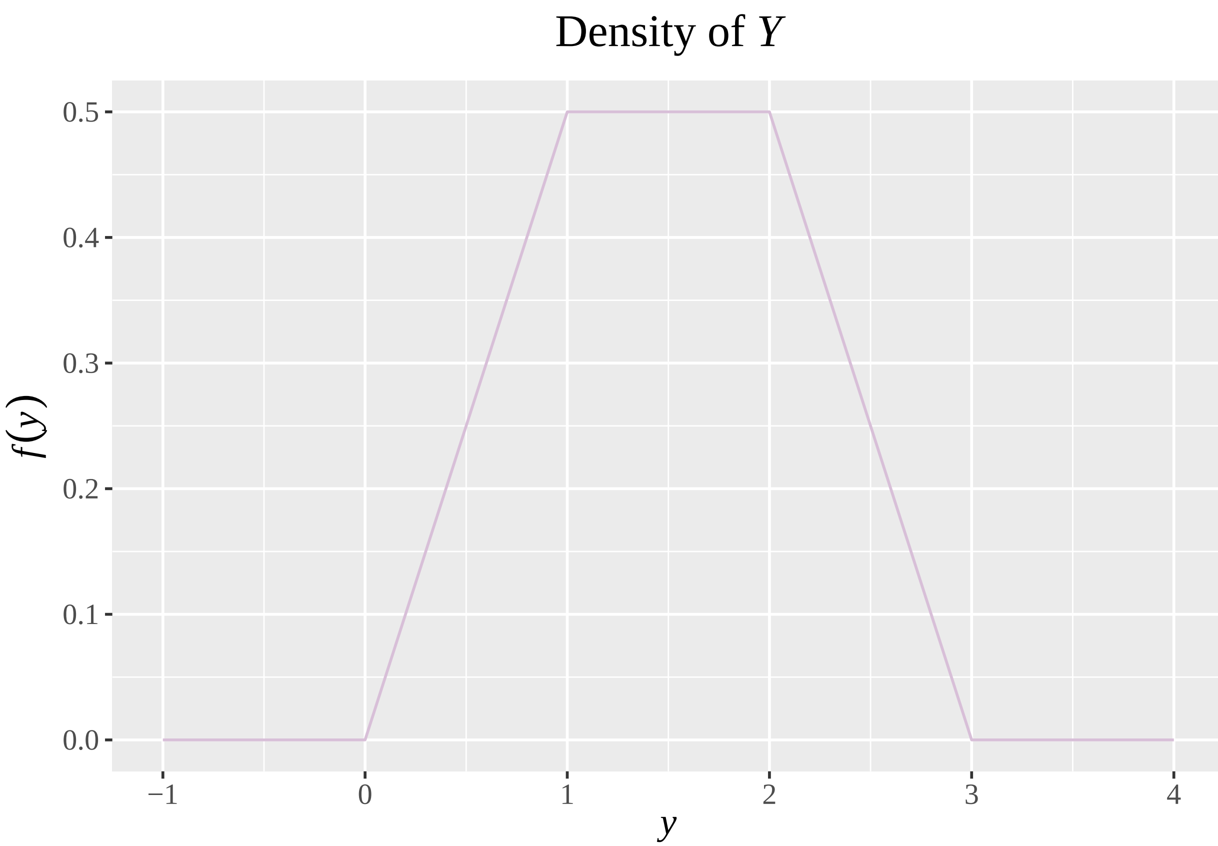
$X_i \sim U(0,1)$. Let $Y = X_1 + 2X_2, Z = X_1$, then$\frac{\partial(x_1,x_2)}{\partial(y,z)} =$

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

$= \frac{-1}{2}$. 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

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

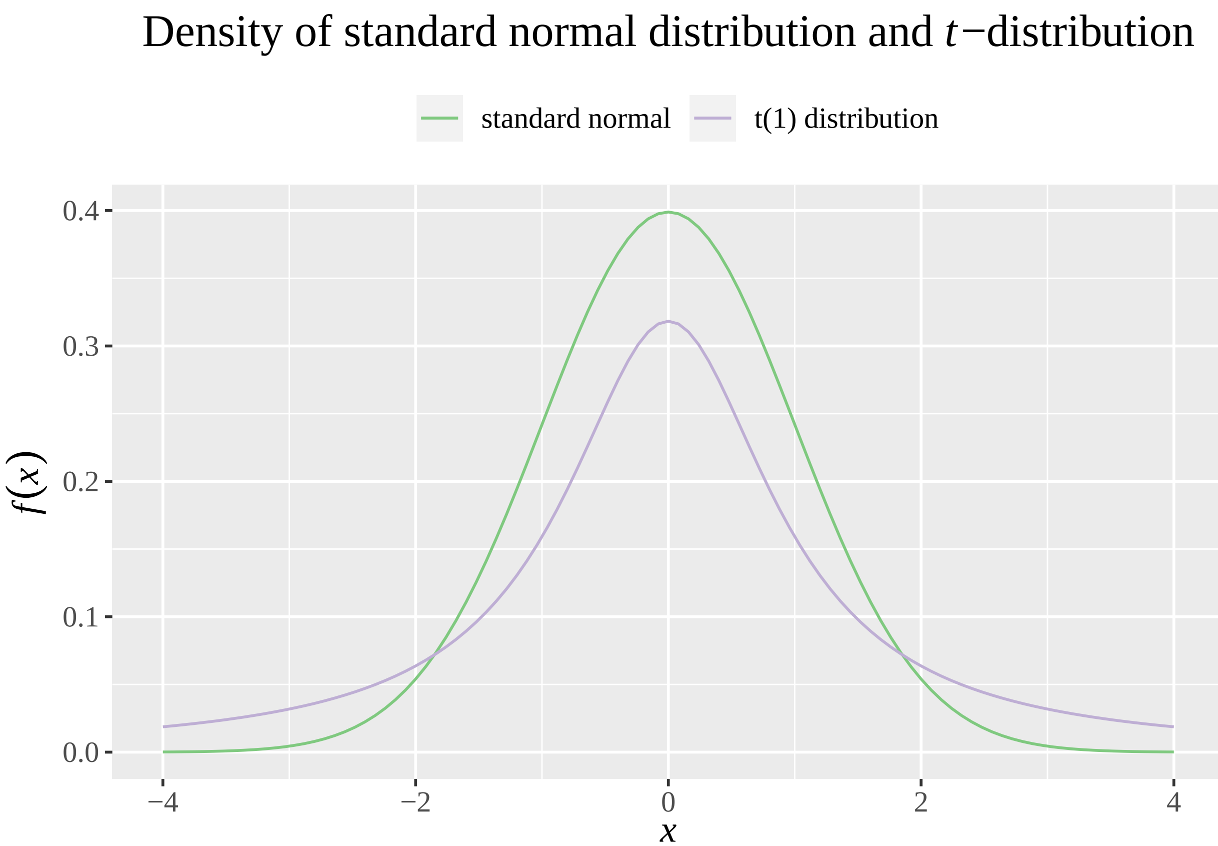
```r
x <- c(-1, 0, 1, 2, 3, 4)
y <- c(0, 0, 0.5, 0.5, 0, 0)
ggplot(data.frame(x, y), aes(x = x, y = y)) +
  geom_line(colour = 'thistle') +
  ggtitle(expression(paste("Density of ", italic(Y)))) +
  theme(plot.title = element_text(hjust = 0.5),
        text = element_text(size = 14, family = "serif"))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

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

```
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
```

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

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

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

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

(f)

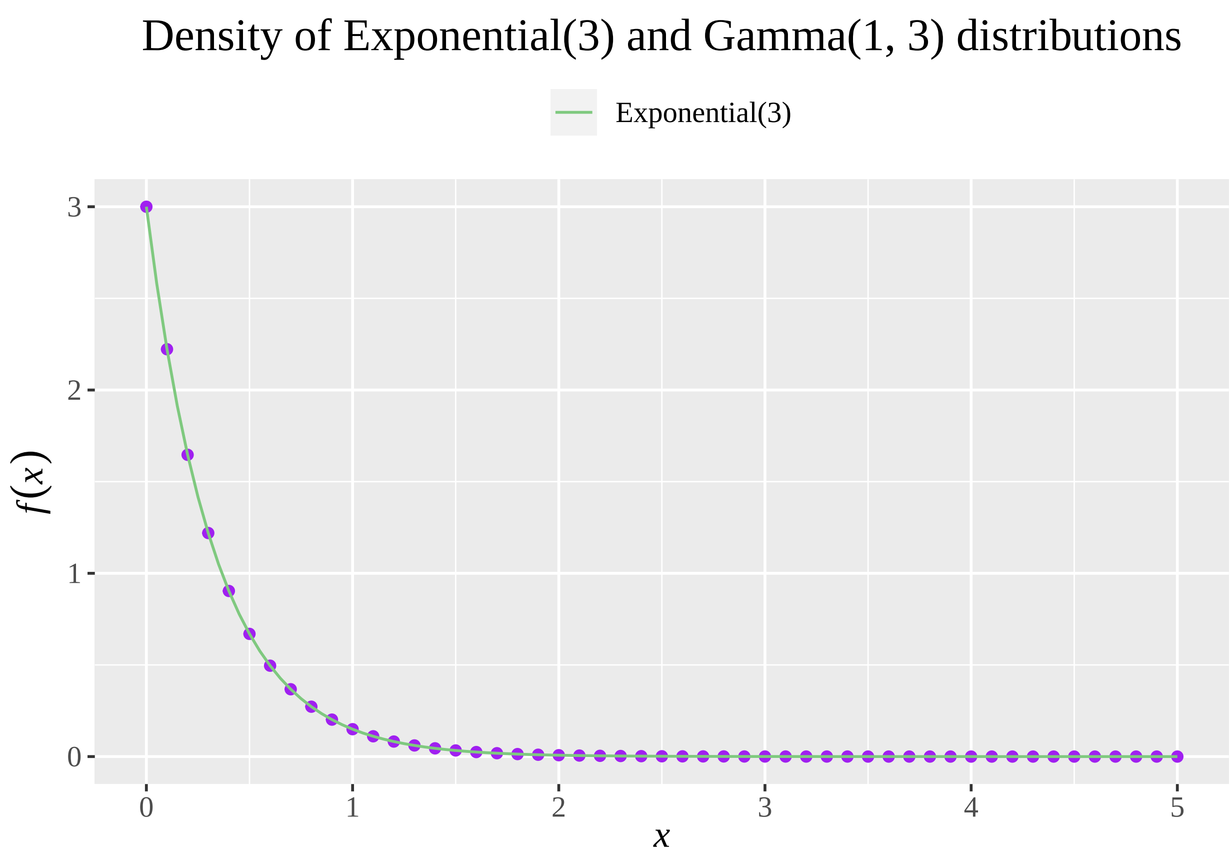
$\Pr(X > 8) \leq 0.4$ is equivalent to $\sum_{k=0}^{8} \Pr(X = k) \geq 0.6$.

```
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)

```
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
```

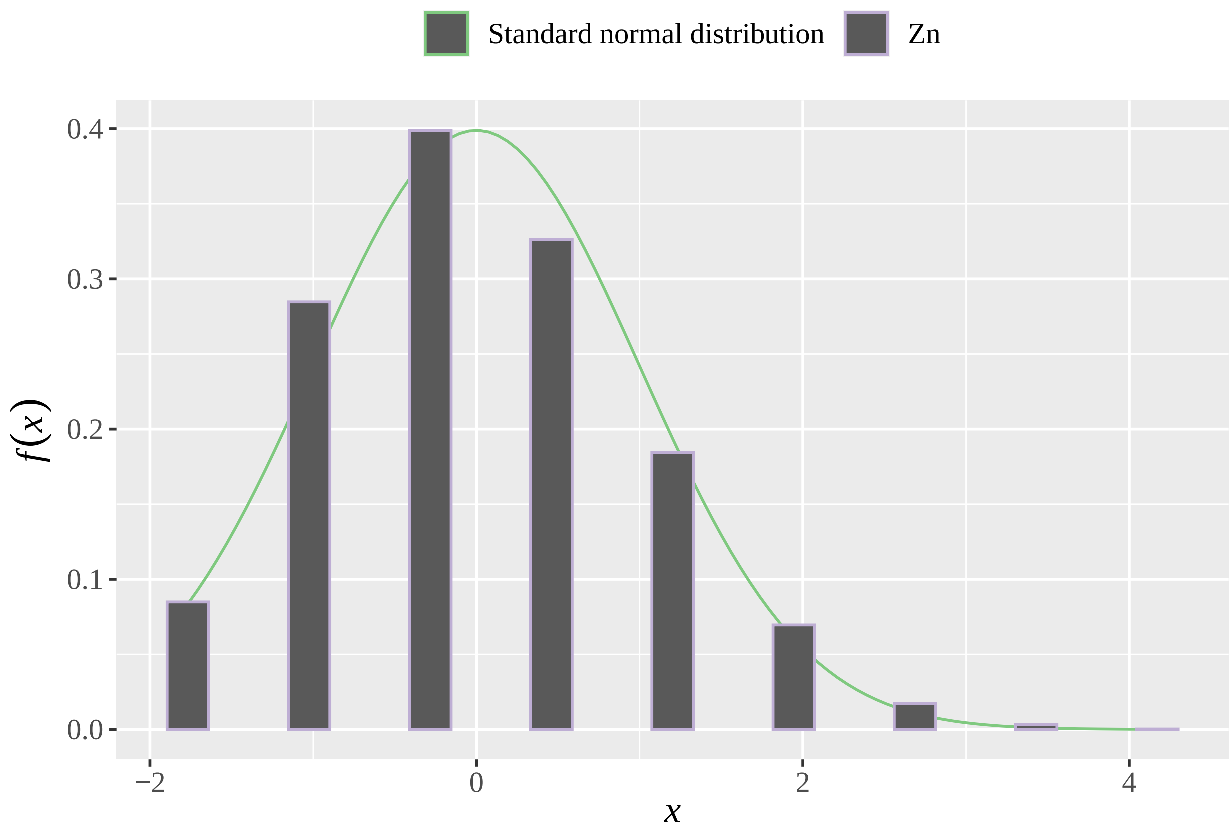
```
## 2      7 0.05
## 3     10 0.10
## 4     12 0.15
## 5     14 0.20
## 6     15 0.25
## 7     17 0.30
## 8     19 0.35
## 9     20 0.40
## 10    22 0.45
## 11    23 0.50
## 12    25 0.55
## 13    27 0.60
## 14    28 0.65
## 15    30 0.70
## 16    32 0.75
## 17    35 0.80
## 18    38 0.85
## 19    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')) +
    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)
}
```

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




```
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
```

```r
n <- c(10, 100, 1000, 10000)
comparision <- data.frame(n, a, b, simulation, true)
comparision
```

```
##       n          a         b simulation        true
## 1    10 -1.1691466 -1.132615     0.0000 0.007515606
## 2   100 -0.5222101  1.206394     0.6304 0.585405139
## 3  1000  1.5303148  2.837857     0.0514 0.060698537
## 4 10000 -3.6357257  1.219524     0.8862 0.888538625
```

– 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     0.7502 0.6826895
## 2   100 -1 1     0.7180 0.6826895
## 3  1000 -1 1     0.6944 0.6826895
## 4 10000 -1 1     0.6862 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)
```

```r
  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 0.1236000 0.2268000 0.2690000 0.2066000 0.1034000 0.03300000 0.01040000
## 2   100 0.1490000 0.2242000 0.2284000 0.1676000 0.0988000 0.04900000 0.01800000
## 3  1000 0.1518000 0.2228000 0.2240000 0.1704000 0.0962000 0.05460000 0.02180000
## 4 10000 0.1440000 0.2246000 0.2212000 0.1638000 0.1076000 0.05180000 0.02200000
## 5  true 0.1493612 0.2240418 0.2240418 0.1680314 0.1008188 0.05040941 0.02160403
##           8           9
## 1 0.001200000 0.000000000
## 2 0.008400000 0.003800000
## 3 0.009800000 0.001000000
## 4 0.009200000 0.003200000
## 5 0.008101512 0.002700504
```

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

```
##    n\\k          1          2         3         4         5         6         7
## 1    10 0.00920000 0.04960000 0.1114000 0.1940000 0.2506000 0.2046000 0.1176000
## 2   100 0.03340000 0.08160000 0.1402000 0.1838000 0.1734000 0.1524000 0.1088000
## 3  1000 0.02780000 0.07940000 0.1442000 0.1720000 0.1800000 0.1482000 0.1066000
## 4 10000 0.03440000 0.08840000 0.1456000 0.1754000 0.1694000 0.1428000 0.1036000
## 5  true 0.03368973 0.08422434 0.1403739 0.1754674 0.1754674 0.1462228 0.1044449
##          8          9
## 1 0.05020000 0.01020000
## 2 0.06140000 0.03160000
## 3 0.06880000 0.03300000
## 4 0.07020000 0.03580000
## 5 0.06527804 0.03626558
```

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

```
##    n\\k           1          2          3          4         5         6
## 1    10 0.000000000 0.00060000 0.00860000 0.03580000 0.1052000 0.2024000
## 2   100 0.005400000 0.01920000 0.04940000 0.07600000 0.1230000 0.1544000
## 3  1000 0.006000000 0.02320000 0.05320000 0.08380000 0.1252000 0.1558000
## 4 10000 0.004600000 0.02180000 0.04680000 0.08760000 0.1350000 0.1442000
## 5  true 0.006383174 0.02234111 0.05212925 0.09122619 0.1277167 0.1490028
##         7         8         9
## 1 0.2550000 0.2420000 0.1250000
## 2 0.1566000 0.1388000 0.1148000
## 3 0.1524000 0.1306000 0.1058000
## 4 0.1558000 0.1330000 0.1002000
## 5 0.1490028 0.1303774 0.1014047
```

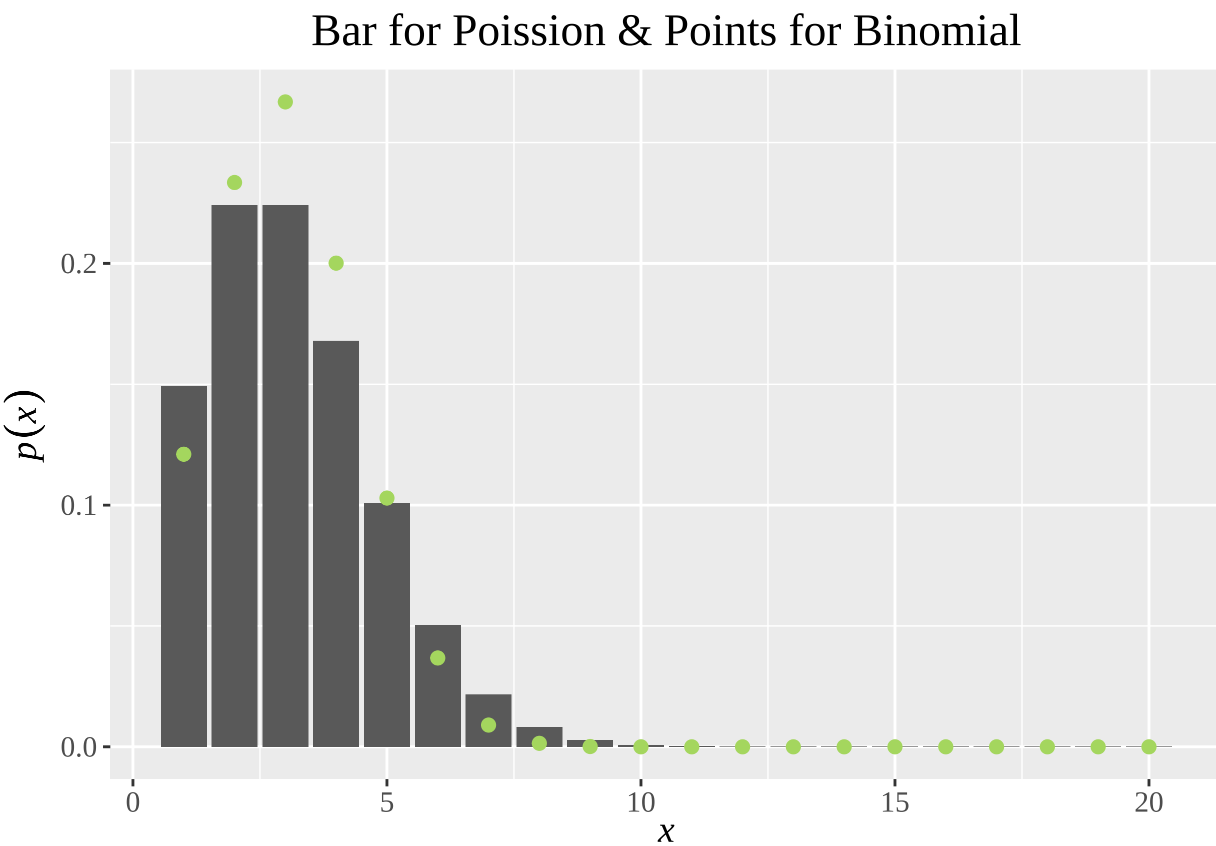
```r
poissonlimit(9, 5000, 9)
```

```
##    n\\k           1           2          3          4          5          6
## 1    10 0.000000000 0.000000000 0.00000000 0.00000000 0.00180000 0.01040000
## 2   100 0.000400000 0.003000000 0.01260000 0.02920000 0.05500000 0.08760000
## 3  1000 0.001400000 0.005800000 0.01100000 0.03320000 0.05980000 0.09300000
## 4 10000 0.002400000 0.004800000 0.01480000 0.03080000 0.05720000 0.09500000
## 5  true 0.001110688 0.004998097 0.01499429 0.03373716 0.06072688 0.09109032
##           7         8         9
## 1 0.0616000 0.1972000 0.3942000
## 2 0.1130000 0.1362000 0.1342000
## 3 0.1220000 0.1304000 0.1272000
## 4 0.1180000 0.1290000 0.1326000
## 5 0.1171161 0.1317556 0.1317556
```

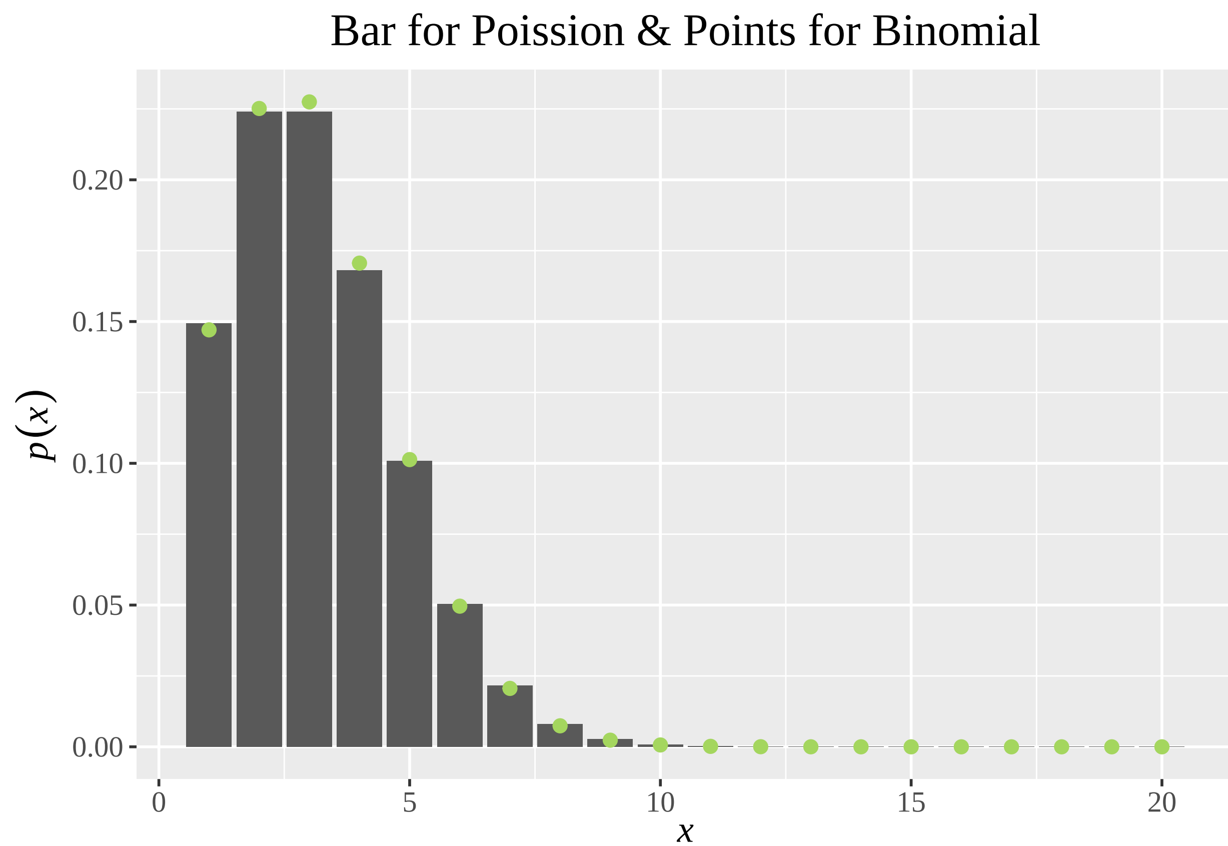
- method 2 (visualization)

```r
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)
}
```

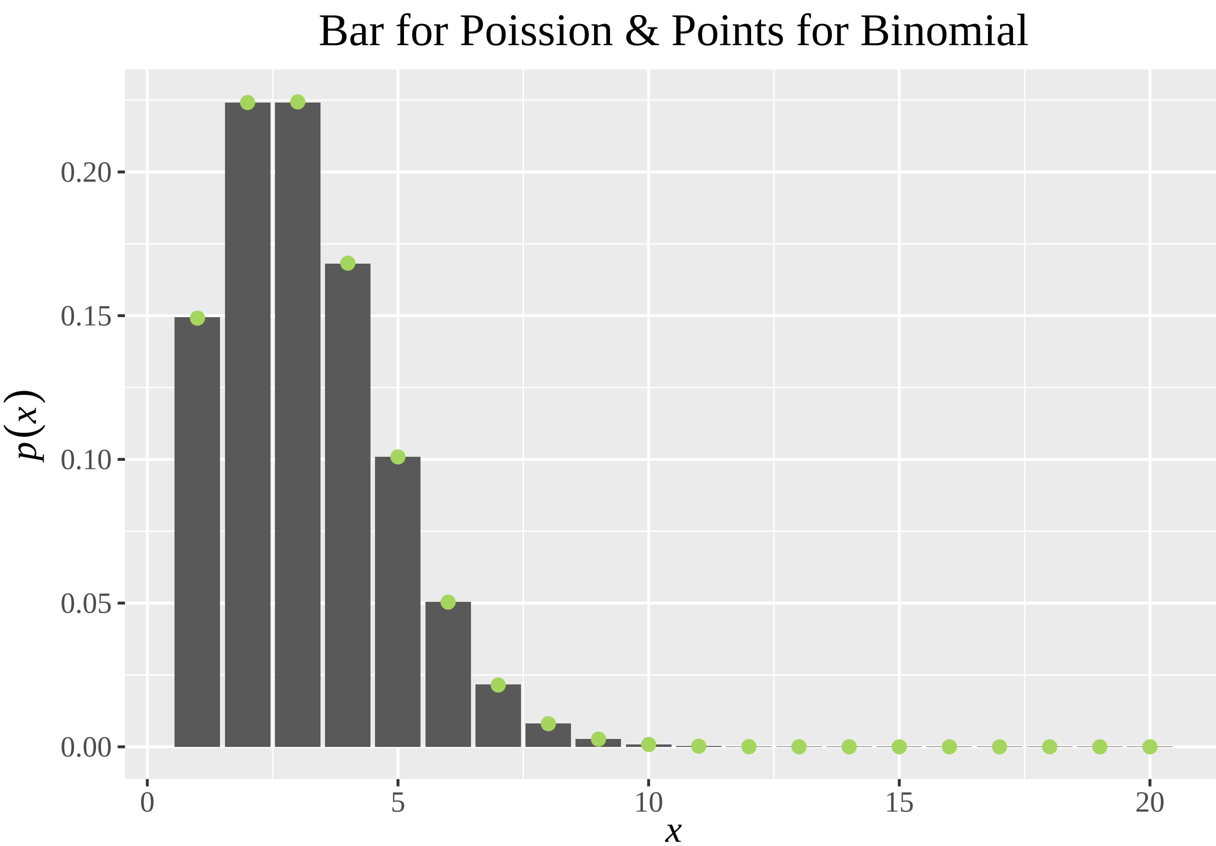
```r
pois_binom(3, 10)
```



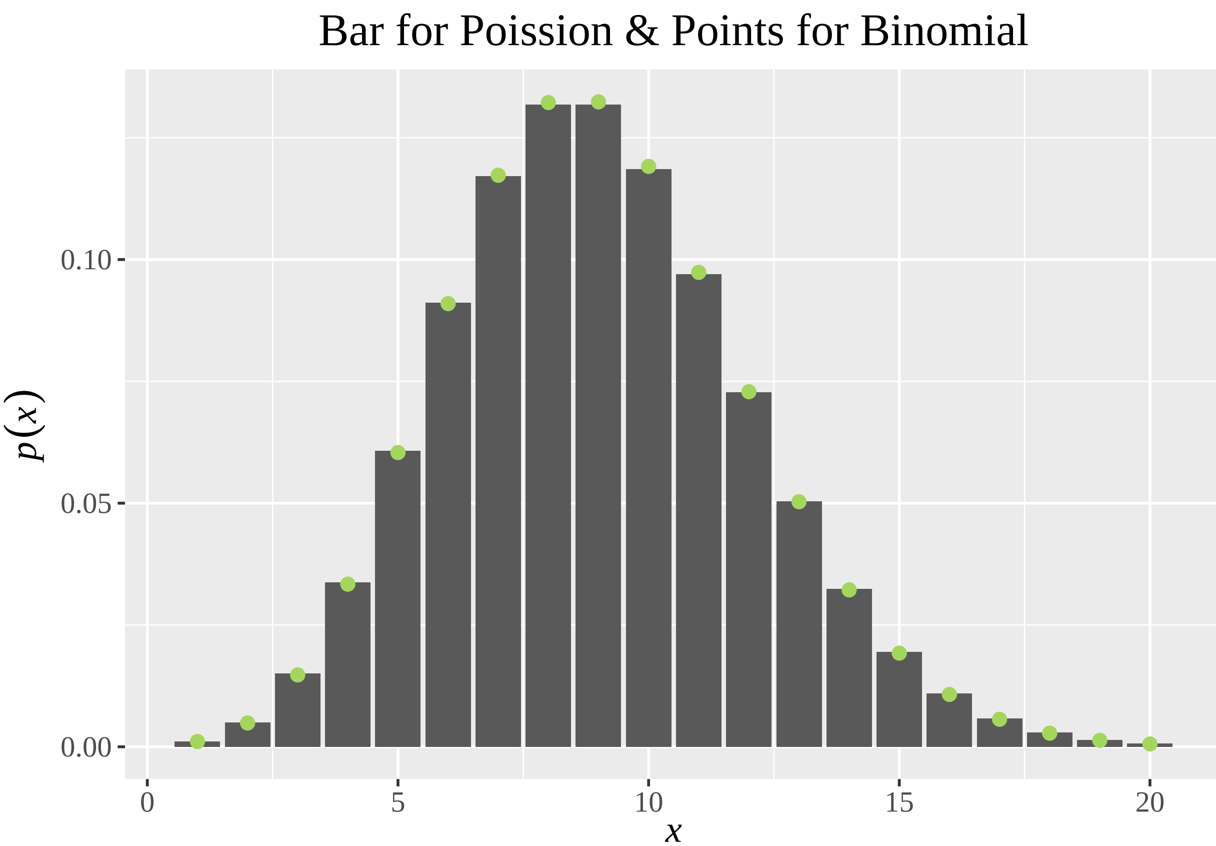

```r
pois_binom(3, 100)
```




```
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') {
```

```r
    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')
    }
  }
}
```

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

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

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

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

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

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

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

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

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

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

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

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

- method 2 (simulation method)

```r
# 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 <-
```

```r
      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") {
      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          0.1286822    0.1290 2 2
## 2          0.2316384    0.2511 2 1
## 3          0.4726027    0.5036 2 0
## 4          0.1326367    0.1252 0 2
## 5          0.1282258    0.1230 1 2
## 6          0.1213592    0.1263 1 2
## 7          0.1318098    0.1281 0 2
## 8          0.1309431    0.1283 2 2
## 9          0.4908000    0.4998 1 0
## 10         0.5229572    0.4953 2 0
```

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

```
##    Pr(X > m+n|X > m) Pr(X > n)  m  n
## 1          0.3461716    0.3515  7  9
## 2          0.2310434    0.2295  1 13
## 3          0.8012889    0.8145 15  1
## 4          0.1499190    0.1541  2 17
## 5          0.5815797    0.5853  3  4
## 6          0.3153289    0.3191  2 10
## 7          0.1309362    0.1324 17 18
## 8          0.2558483    0.2549  1 12
## 9          0.2646922    0.2579  7 12
## 10         0.1952586    0.1866 13 15
```

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

```
##    Pr(X > m+n|X > m) Pr(X > n)   m   n
## 1          0.8633841    0.8570  57  14
## 2          0.9088807    0.9040 151   9
## 3          0.1769373    0.1792   2 172
## 4          0.7494530    0.7613  98  27
## 5          0.7291635    0.7284  40  31
## 6          0.4552106    0.4510  76  76
```

```
## 7          0.1521036    0.1582 185 182
## 8          0.4995624    0.4991   8  67
## 9          0.1878223    0.1885  35 166
## 10         0.5963842    0.5990  13  49
```

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

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

```
##    Pr(X > m+n|X > m) Pr(X > n) m n
## 1         1.00000000    0.6314 2 0
## 2         0.25064599    0.2605 2 1
## 3         0.07317073    0.0771 1 2
## 4         1.00000000    0.6415 2 0
## 5         0.31008340    0.2638 1 1
## 6         1.00000000    0.6306 0 0
## 7         0.30313326    0.2649 1 1
## 8         0.06825811    0.0806 1 2
## 9         0.29725551    0.2587 1 1
## 10        0.04298357    0.0791 2 2
```

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

```
##    Pr(X > m+n|X > m) Pr(X > n) m n
## 1        0.083527886    0.5574 5 4
## 2        0.180845511    0.7321 5 3
## 3        0.095770805    0.3860 3 5
## 4        0.002202643    0.1362 7 7
## 5        0.017497933    0.1300 3 7
## 6        0.005507474    0.1297 5 7
## 7        0.174250681    0.5516 3 4
## 8        0.057784912    0.3791 4 5
## 9        0.004126902    0.1311 5 7
## 10       0.014874739    0.2367 5 6
```

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

```
##    Pr(X > m+n|X > m) Pr(X > n)  m  n
## 1       0.0031055901    0.3113  9 11
## 2       0.0000000000    0.1314 10 13
## 3       0.0183930300    0.7879 12  7
## 4       0.0004777831    0.4210 12 10
## 5       0.0093868282    0.4116  8 10
## 6       0.0003694809    0.1427  9 13
## 7       0.0022404780    0.6654 13  8
## 8       0.0040753948    0.2139  7 12
## 9       0.0023171135    0.4201 11 10
## 10      0.0065446973    0.3080  8 11
```

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

```
##    Pr(X > m+n|X > m) Pr(X > n)  m  n
## 1                 0    0.4654 43 50
## 2                 0    0.5723 48 48
## 3                 0    0.4619 50 50
## 4                 0    0.4702 43 50
## 5                 0    0.3541 46 52
## 6                 0    0.5812 52 48
## 7                 0    0.3549 43 52
## 8                 0    0.5779 53 48
## 9                 0    0.3071 55 53
## 10                0    0.3506 47 52
```

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