Chapter 11

(6) Prove parts (b) and (c) (b) $\widehat{T_r c_k} = u_n^{-rk}\hat{c}_k$.

$$\begin{aligned}
\widehat{T_r c_k} &= \frac{1}{n}\Sigma_0^{n-1} T_r c_j e^{-2\pi ijk/n} \\
&\overset{(*)}{=} \frac{1}{n}\Sigma_0^{n-1} c_{j-r} e^{-2\pi ijk/n} \\
&= \frac{1}{n}\Sigma_0^{n-1} c_{j-r} e^{-2\pi i(j-r)k/n} e^{-2\pi irk/n} \\
&\overset{(**)}{=} u_n^{-rk}\hat{c}_k
\end{aligned}$$

(*) holds because $T_r c_j = c_{j-r}$ (**) holds because $\hat{c}_{j-r} = \frac{1}{n}\sum_{j=0}^{n-1} c_{j-r} e^{-2\pi i\frac{(j-r)k}{n}}$ and $u_n = e^{2\pi i/n}$

(c) $\widehat{Rc_k} = R\hat{c}_k$.

$$\begin{aligned}
\widehat{Rc_k} &= \frac{1}{n}\sum_{j=0}^{n-1} Rc_j e^{-2\pi i\frac{jk}{n}} \\
&\overset{(*)}{=} \frac{1}{n}\sum_{j=0}^{n-1} c_{-j} e^{-2\pi i\frac{jk}{n}} \\
&= R\hat{c}_k
\end{aligned}$$

(*) holds because $Rc_j = c_{-j}$

(19) The $n \times n$ permutation matrix $\boldsymbol{S}_n$ satisfying $\boldsymbol{S}_n\boldsymbol{e}_k = \boldsymbol{e}_{k+1}$ for $k = 1, \ldots, n-1$ and $\boldsymbol{S}_n\boldsymbol{e}_n = \boldsymbol{e}_1$ is called a circular shift. Prove that the $n \times n$ matrix C is circulant if and only if $S_nC = CS_n$.
Solution : the ijth entry of circular shift S is given by $S_{ij} = 1(i = j+1)$ In particular, the left (right) multiplication by S equals to row (column) circular permutation,
so for any circulant matrix C we can have $SC = CS$ note we have

$$(SC)_{ij} = \sum_{\ell} S_{i\ell}C_{\ell j} = C_{i-1,j}$$

$$(WC)_{ij} = \sum_{\ell}^{\ell} C_{i\ell}S_{\ell i} = C_{i,j+1}$$

Then we get

$$i-1, j = C_{i,j+1} \Leftrightarrow C_{i,j} = C_{i-1,j-1} \Leftrightarrow C_{ij} = C_{i+k,j+k}$$

Hence the matrix C is a circulant matrix
Proved

(20) Prove that the sum or product of two circulant matrices of the same size is circulant. Also prove that the inverse of an invertible circulant matrix C is circulant. Thus, the collection of invertible circulant matrices forms a group. Why is this group commutative? *Solution* : A matrix is circulant if and only if it commutes with

$$J = \begin{pmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \\ 1 & 0 & 0 & \cdots & 0 \end{pmatrix}$$

So the sum or product of two circulant matrices of the same size is circulant too. By Cayley-Hamilton theorem, the inverse of every nonsingular matrix A can be expressed as a polynomial in A of degree $\leq n-1$. Therefore, if A is in the linear span of $I, J, \ldots, J^{n-1}$, then A^{-1} is a polynomial in J of degree $\leq (n-1)^2$. Since $J^n = I$, it follows that A^{-1} lies inside the linear span of $I, J, \ldots, J^{n-1}$.
Hence it is circulant. Circulant matrices commute because they are simultaneously diagonaliz-able.

Chapter 12

```
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 1.4.1
## v readr   2.1.3      v forcats 0.5.2
## v purrr   0.3.4
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(gridExtra)
```

```
##
## Attaching package: 'gridExtra'
##
## The following object is masked from 'package:dplyr':
##
##     combine
```

```
library(grid)
library(MASS)
```

```
##
## Attaching package: 'MASS'
##
## The following object is masked from 'package:dplyr':
##
##     select
```

(1) Discuss how you would use the inverse method to generate a random variable with (a) the continuous logistic density

$$f(x \mid \mu, \sigma) = \frac{e^{-\frac{x-\mu}{\sigma}}}{\sigma\left[1 + e^{-\frac{x-\mu}{\sigma}}\right]^2}$$

(b) the Pareto density

$$f(x \mid \alpha, \beta) = \frac{\beta\alpha^\beta}{x^{\beta+1}} 1_{(\alpha,\infty)}(x),$$

and (c) the Weibull density

$$f(x \mid \delta, \gamma) = \frac{\gamma}{\delta} x^{\gamma-1} e^{-\frac{x^\gamma}{\delta}} 1_{(0,\infty)}(x),$$

where $\alpha, \beta, \gamma, \delta$, and σ are taken positive.
Solution :
Step 1: Generate u from uniform $(0, 1)$
Step 2: Return $x = F_X^{-1}(u)$

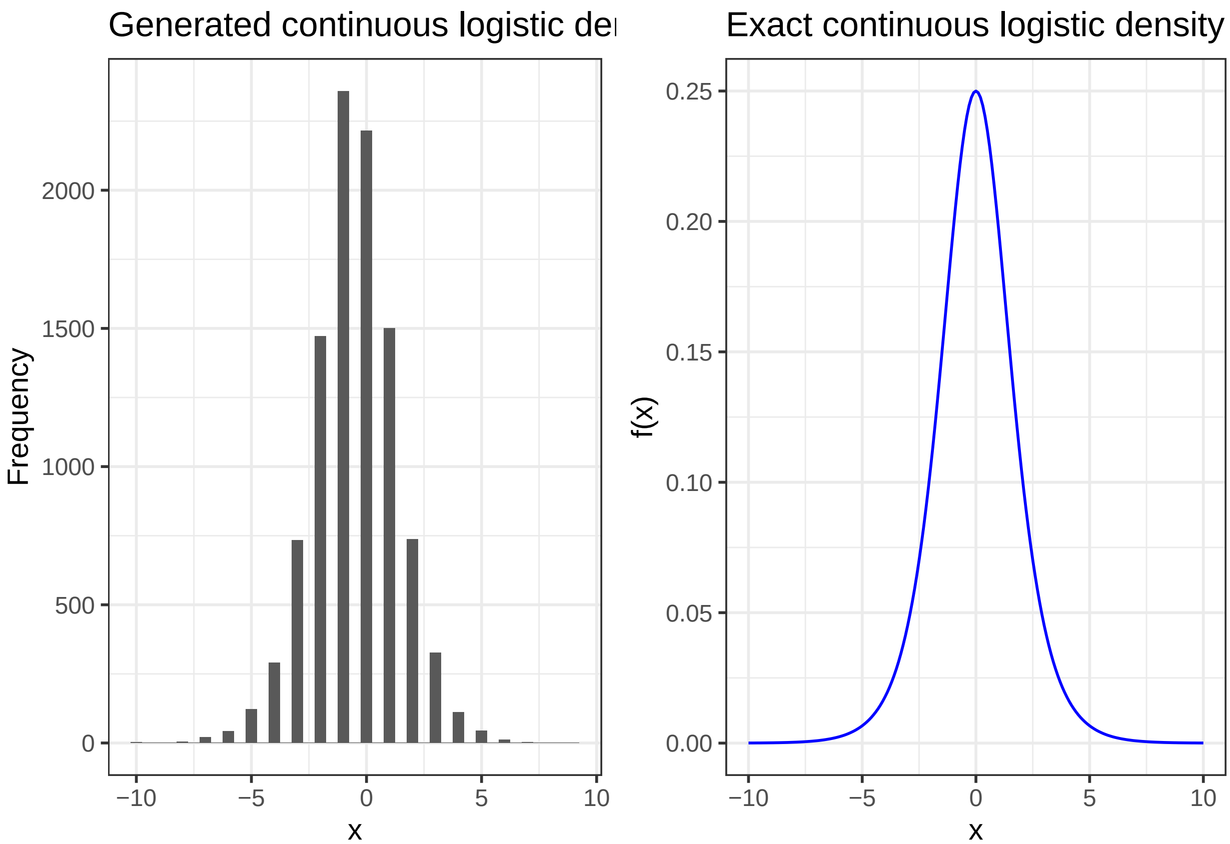
```
# generate 10,000 random uniform variables
set.seed(1)
u <- runif(10000)
```

(a) continuous logistic density

$$F_X(x) = \frac{1}{1 + e^{-\frac{x-\mu}{\sigma}}} = u$$
$$\frac{1}{u} = 1 + e^{-\frac{x-\mu}{\sigma}}$$
$$\frac{1-u}{u} = e^{-\frac{x-\mu}{\sigma}}$$
$$\log(\frac{1-\mu}{\mu}) = \frac{\mu - x}{\sigma}$$
$$F_X^{-1} = x = \mu - \log(\frac{1-u}{u}) \cdot \sigma$$

Take an example with $\mu = 0$ and $\sigma = 1$

```
mu = 0
sigma = 1
generated_vals <- floor( mu - log( (1-u)/u )*sigma )
plts <- list()


x_dlogis <- seq(- 10, 10, by = 0.1)
y_dlogis <- dlogis(x_dlogis,location = 0, scale = 1)

# construct histogram of generated values
plts[[1]] <- tibble(generated_vals = generated_vals) %>%
  ggplot(aes(x = generated_vals)) +
        geom_histogram(binwidth = 0.5 ) +
        theme_bw() +
        labs(x = "x", y = "Frequency", title = "Generated continuous logistic density")

plts[[2]] <- tibble(x = x_dlogis, theoretical_pdf = y_dlogis) %>%
    ggplot() +
        geom_line(aes(x = x, y = theoretical_pdf), colour = "blue") +
        theme_bw() +
        labs(x = "x", y = "f(x)", title = "Exact continuous logistic density")

grid.arrange(grobs = plts, nrow = 1)
```



(b) Pareto density

$$F_X(x) = 1 - \left(\frac{\alpha}{x}\right)^{\beta}; x \geq \alpha$$

$$F(X) = U$$

$$U \sim \text{ Uniform } (0, 1)$$

$$1 - \left(\frac{\alpha}{x}\right)^{\beta} = u$$

$$F_X^{-1} = x = \alpha(1-u)^{-1/\beta}$$

(c) Weibull density

$$\begin{aligned}
F_X(x) = u = 1 - \exp\left(\frac{x^{\gamma}}{\delta}\right), \text{ where x} \geq 0 \\
\implies \log(1-u) = \frac{x^{\gamma}}{\delta} \\
\implies \log(1-u)\delta = x^{\gamma}. \\
F_X^{-1} = x = (\log(1-u)\delta)^{1/\gamma} \\
= \log(1-u)^{1/\gamma} \cdot (\delta)^{1/\gamma}
\end{aligned}$$

(2) Continuing problem 1, discuss how the inverse method applies to (d) the Gumbel density

$$f(x) = e^{-x}e^{-e^{-x}},$$

(e) the arcsine density

$$f(x) = \frac{1}{\pi\sqrt{x(1-x)}}1_{(0,1)}(x),$$

and (f) the slash density

$$f(x) = \alpha x^{\alpha-1}1_{(0,1)}(x),$$

where $\alpha > 0$.

Solution :
Step 1: Generate u from uniform $(0,1)$;
Step 2: Return $x = F_X^{-1}(u)$
(d) Gumbel density

$$\begin{aligned} F_X(x) &= e^{-e^{(-x)}} = u \\ -e^{(-x)} &= \log(u) \\ -x &= \log(-\log(u)) \\ F_X^{-1} &= x = -\log(-\log(u)) \end{aligned}$$

(e) arcsine density

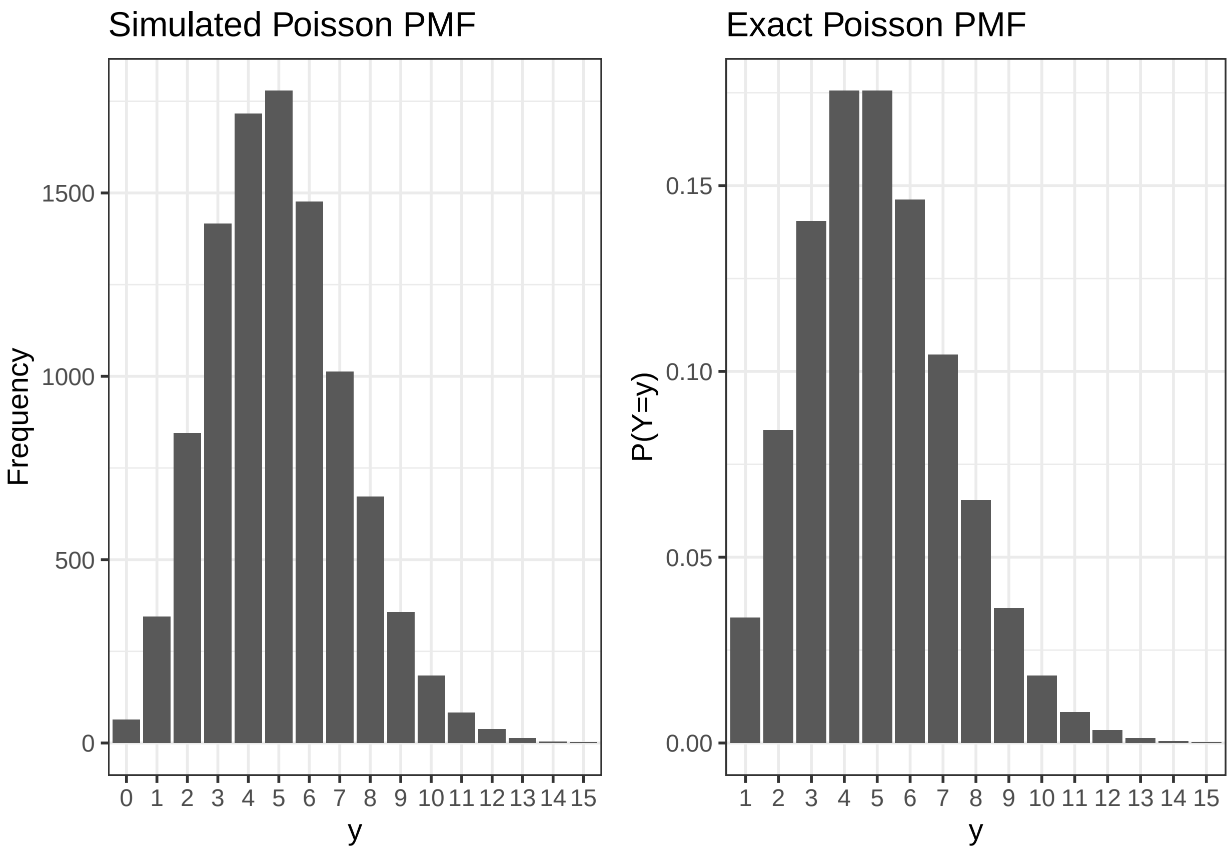
$$\begin{aligned} F_X(x) &= \frac{2}{\pi}\arcsin(\sqrt{x}) = u \\ \arcsin(\sqrt{x}) &= u \cdot \frac{\pi}{2} \\ F_X^{-1} &= x = \sin^2(u \cdot \frac{\pi}{2}) \end{aligned}$$

(f) slash density The slash density I found on the Internet is different from here.

$$\begin{aligned} F_X(x) = \int_0^x f(t)dt &= \int_0^x \alpha t^{\alpha-1}dt = x^\alpha = u \\ F_X^{-1} &= x = u^{1/\alpha} \end{aligned}$$

(5) Describe one method for generating independent Poisson deviates and implement it in code.
Solution :
$Y \sim \text{Pois}(\lambda)$
1. Define parameter λ =5.
2. Generate k random uniform variables $U \sim Unif[0,1]$.
3. Derive an expression for y.
The probability mass function is $P\{Y = y\} = \frac{e^{-\lambda}\lambda^y}{y!}$ for $y = 0, 1, 2, \ldots$
We can recursively define this with $p_0 = e^{-\lambda}$ and $p_{i+1} = \frac{\lambda}{i+1}p_i$.
The cumulative distributive function can be expressed as $F(y) = \sum_{i=0}^{y} p_i$.
We want y to be the value of j which $\sum_{i=0}^{j-1} p_i \leq u < \sum_{i=0}^{j} p_i$.
4. We define a function that uses the recursive expression of p_i to build the CDF until we obtain y.
For the base case $i = 0$, we define p_0 and initiate $F(y)$. We increment i and update $F(y)$ with the new p_i until we find y.

```r
#Define parameter
lambda =5
#Generate random uniform variables
u <- runif(10000)
#set function
poisgen = function(u){
    y = 0
    i = 0
    p = exp(-lambda)
    F = p
    while(u>=F){
        p = (p*lambda)/(i+1)
        F = F+p
        i = i+1
        y = i
    }
    return(y)
}
ysim <- sapply(u, poisgen)

pois <- dpois(1:15, lambda)

# create list to hold our two plots
plts <- list()

plts[[1]] <- tibble(sim=ysim) %>% ggplot() +
  geom_bar(aes(x=as.factor(sim)), stat="count") +
  labs(x="y", y="Frequency", title="Simulated Poisson PMF") +theme_bw()

plts[[2]] <- tibble(x=1:15, pmf=pois) %>% ggplot() +
  geom_bar(aes(x=as.factor(x), y=pmf), stat="identity") +
  labs(x="y", y="P(Y=y)", title="Exact Poisson PMF") +theme_bw()

grid.arrange(grobs = plts, nrow = 1)
```



(6) Describe and implement acceptance-rejection sampling for the beta distribution with parameters $\alpha > 1$ and $\beta > 1$. In this case $x^{\alpha-1}(1-x)^{\beta-1} \leq c$ for all $x \in [0,1]$ and some constant c. First, calculate the least value of c.

Generate $Y \sim \text{Beta}(\alpha = 2.7, \beta = 6.3)$

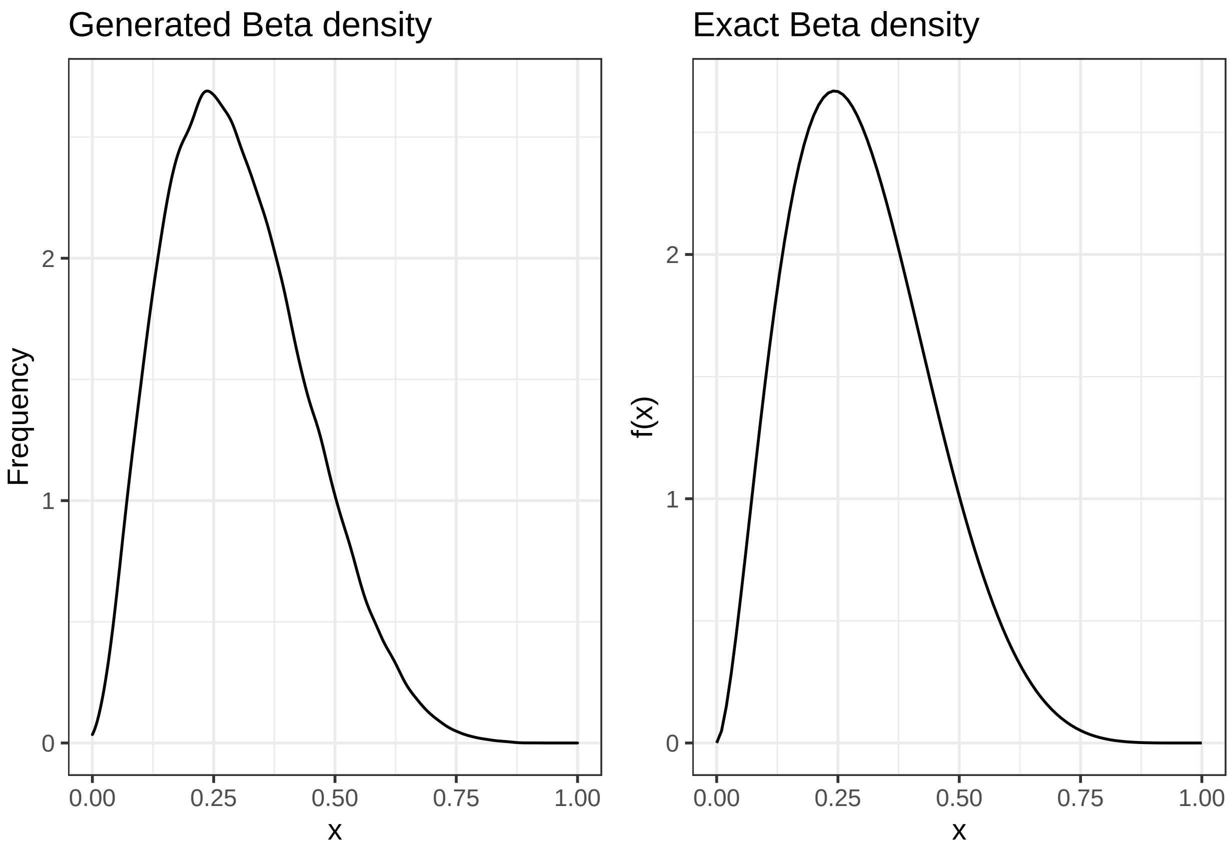
1. Generate (U, V) independent uniform $(0, 1)$.
2. If $U < \frac{1}{c} f_Y(V)$ then, set $Y = V$; otherwise, return to step 1. Where $c = \max_y f_Y(y) = 2.669$

```
f1 = function(x){
  a = 2.7
  b = 6.3
  beta = gamma(a) * gamma(b) /gamma(a + b)
  p = x ** (a - 1) * (1-x) ** (b - 1)
  return (1/beta * p)
}
mode = (2.7-1)/(2.7+6.3-2) ##Mode of Beta distribution
c = f1(mode)
print(c)
```

```
## [1] 2.669744
```

```
beta_gen = function(n){
  set.seed(199)
  i = 0
  output = c()
  while (i < n){
    U = runif(1)
```

```
    V = runif(1)
    if( U < 1/c * f1(V)){
      output[i] = V
      i = i + 1
  }
  }
  return (output)
}

betagen = beta_gen(n = 100000)

plts <- list()

plts[[1]] <- tibble(sim = betagen) %>%
  ggplot() + geom_density(aes(x = sim)) + labs(x = "x", y = "Frequency", title =
                                                 "Generated Beta density") +
  xlim(0,1)  + theme_bw()

gam <- dbeta(seq(0, 1,0.01), shape1 = 2.7,shape2 = 6.3,ncp = 0)
# construct line chart of exact pdf
plts[[2]] <- tibble(x = seq(0, 1,0.01), pdf = gam) %>%
  ggplot() + geom_line(aes(x = x, y = pdf)) + labs(x = "x", y = "f(x)", title =
                                                     "Exact Beta density") +
  xlim(0, 1)  + theme_bw()

#arrange two plots
grid.arrange(grobs = plts, nrow = 1)
```


Generated Beta density
Frequency
0
1
2
0.00
0.25
0.50
0.75
1.00
x
Exact Beta density
f(x)
0
1
2
0.00
0.25
0.50
0.75
1.00
x