Assignment 4

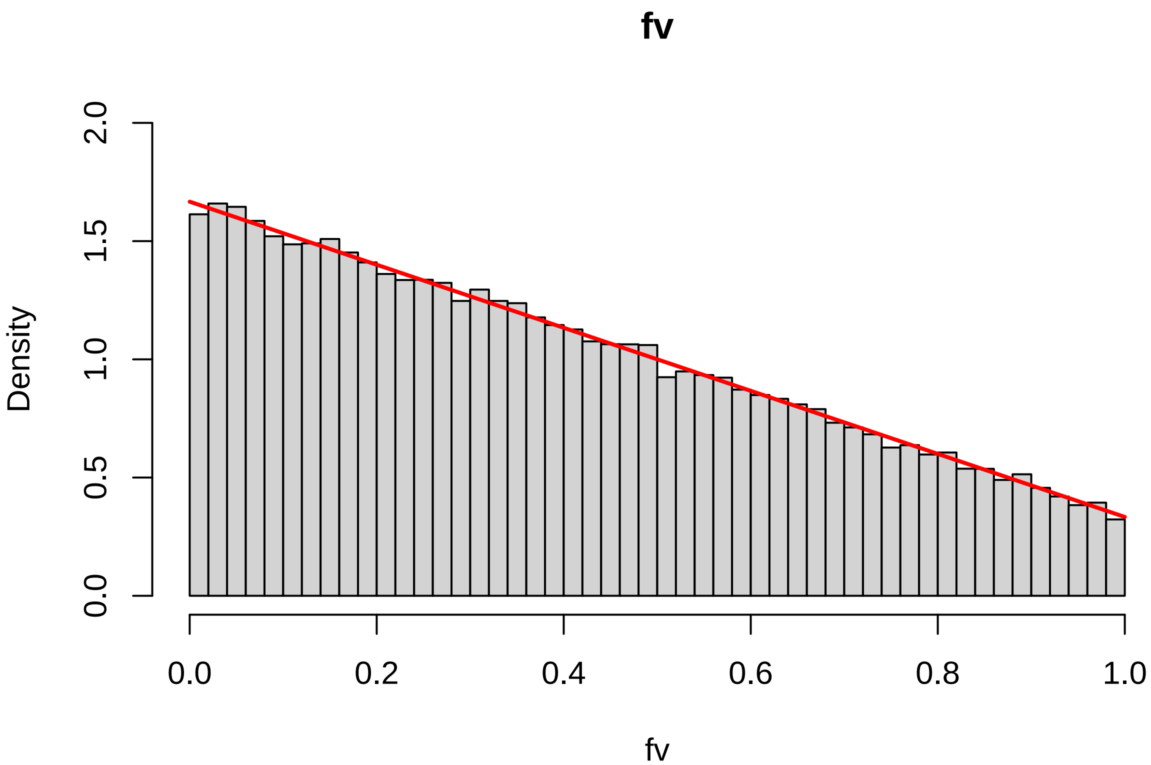
```
personal_number <- 180
```

1.

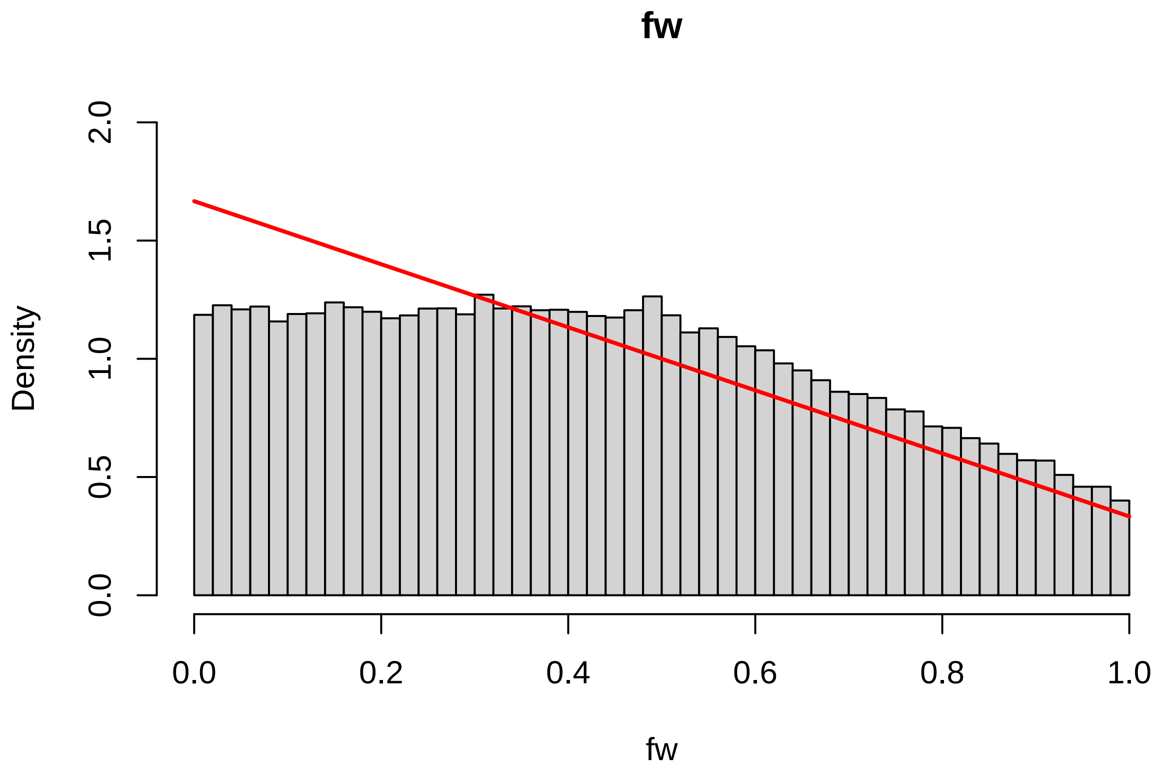
```
f <- function(x) {
  (5/3) * (1 - 0.8 * x)
}
# (density without knowing c)
g <- function(x) {
  (1 - 0.8 * x)
}
N <- 100000

set.seed(personal_number)

# f(x) with v_i
fv <- numeric(N)
i <- 1
while (i <= N) {
  u_i <- runif(1, 0, 1)
  v_i <- runif(1, 0, 5/3)
  if (f(u_i) >= v_i) {
    fv[i] <- u_i
    i <- i + 1
  }
}

set.seed(personal_number)

# f(x) with w_i
fw <- numeric(N)
i <- 1
while (i <= N) {
  u_i <- runif(1, 0, 1)
  w_i <- runif(1, 0, 1)
  if (f(u_i) >= w_i) {
    fw[i] <- u_i
    i <- i + 1
  }
}

set.seed(personal_number)

# f(x) with z_i
fz <- numeric(N)
```

```r
i <- 1
while (i <= N) {
  u_i <- runif(1, 0, 1)
  z_i <- runif(1, 1/3, 5/3)
  if (f(u_i) >= z_i) {
    fz[i] <- u_i
    i <- i + 1
  }
}

set.seed(personal_number)

# g(x) with v_i
gv <- numeric(N)
i <- 1
while (i <= N) {
  u_i <- runif(1, 0, 1)
  v_i <- runif(1, 0, 5/3)
  if (g(u_i) >= v_i) {
    gv[i] <- u_i
    i <- i + 1
  }
}

set.seed(personal_number)

# g(x) with w_i
gw <- numeric(N)
i <- 1
while (i <= N) {
  u_i <- runif(1, 0, 1)
  w_i <- runif(1, 0, 1)
  if (g(u_i) >= w_i) {
    gw[i] <- u_i
    i <- i + 1
  }
}

set.seed(personal_number)

# g(x) with w_i
gz <- numeric(N)
i <- 1
while (i <= N) {
  u_i <- runif(1, 0, 1)
  z_i <- runif(1, 1/3, 5/3)
  if (g(u_i) >= z_i) {
    gz[i] <- u_i
    i <- i + 1
  }
}

x_vals <- seq(0, 1, length.out = 1000)
```

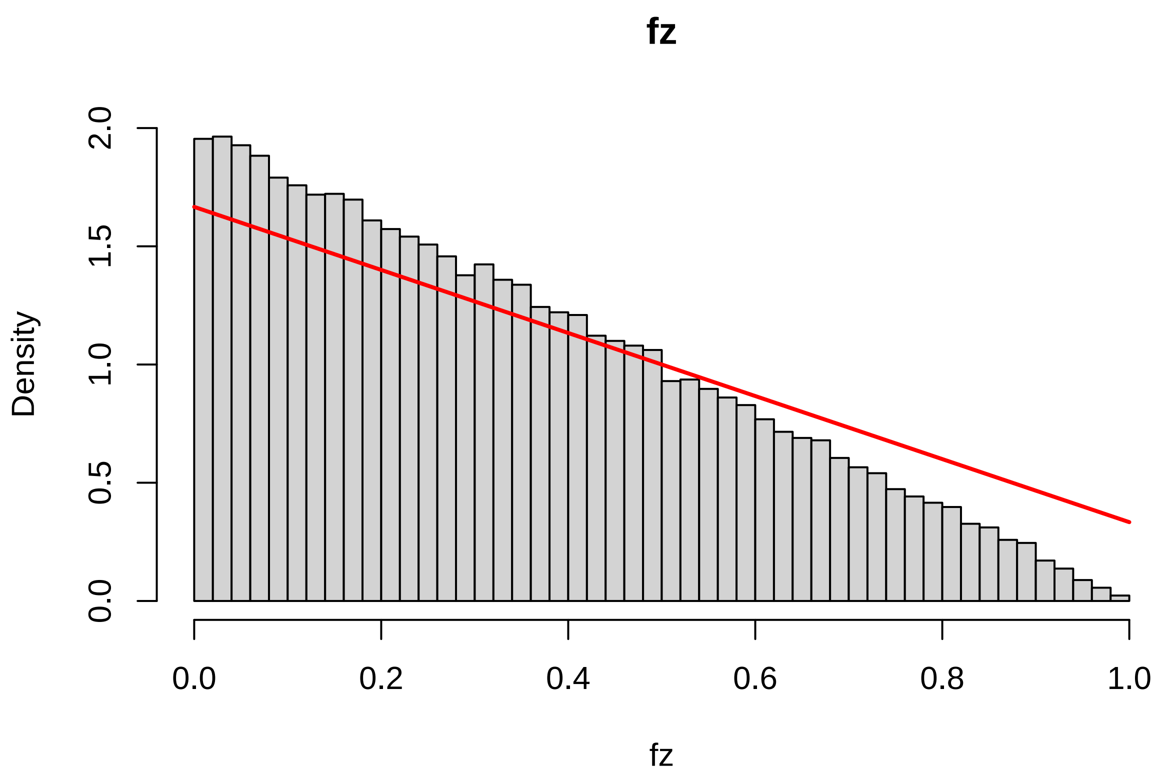
```r
f_density <- f(x_vals)
g_density <- g(x_vals)

hist(fv, breaks = 50, probability = TRUE, main = "fv", ylim = c(0, 2), xlim = c(0, 1))
lines(x_vals, f_density, col = "red", lwd = 2)
```



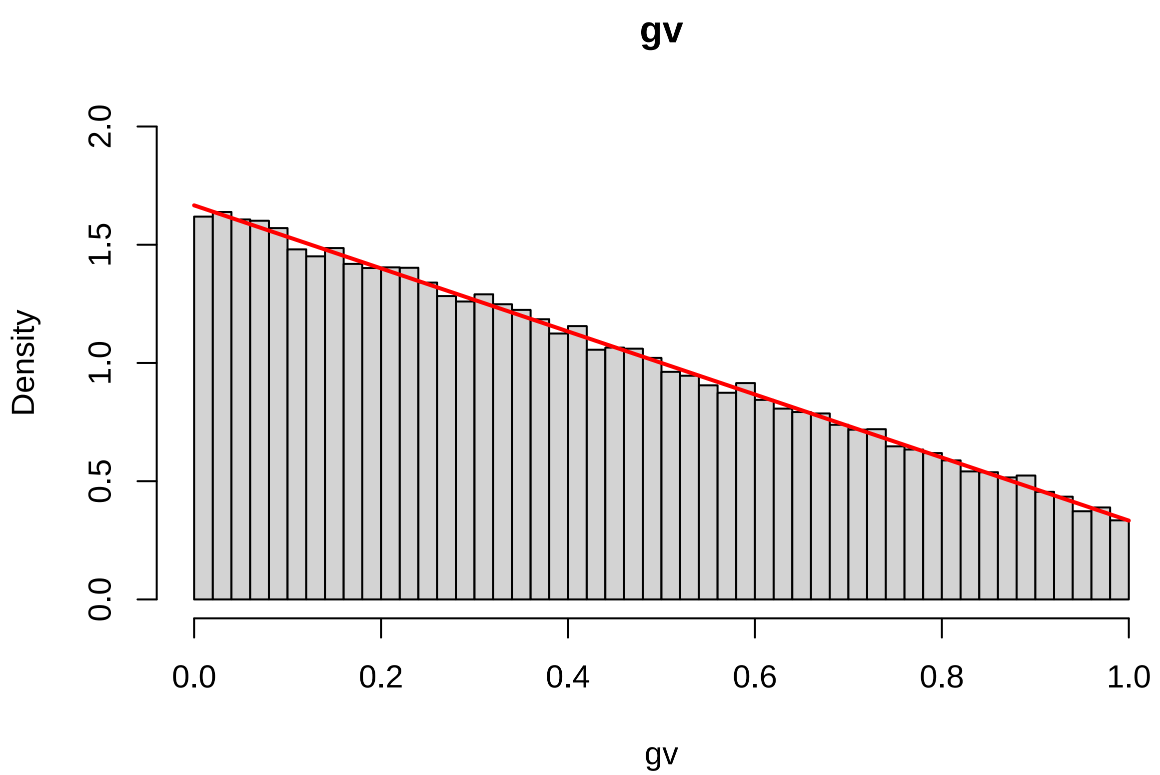

```r
hist(fw, breaks = 50, probability = TRUE, main = "fw", ylim = c(0, 2), xlim = c(0, 1))
lines(x_vals, f_density, col = "red", lwd = 2)
```



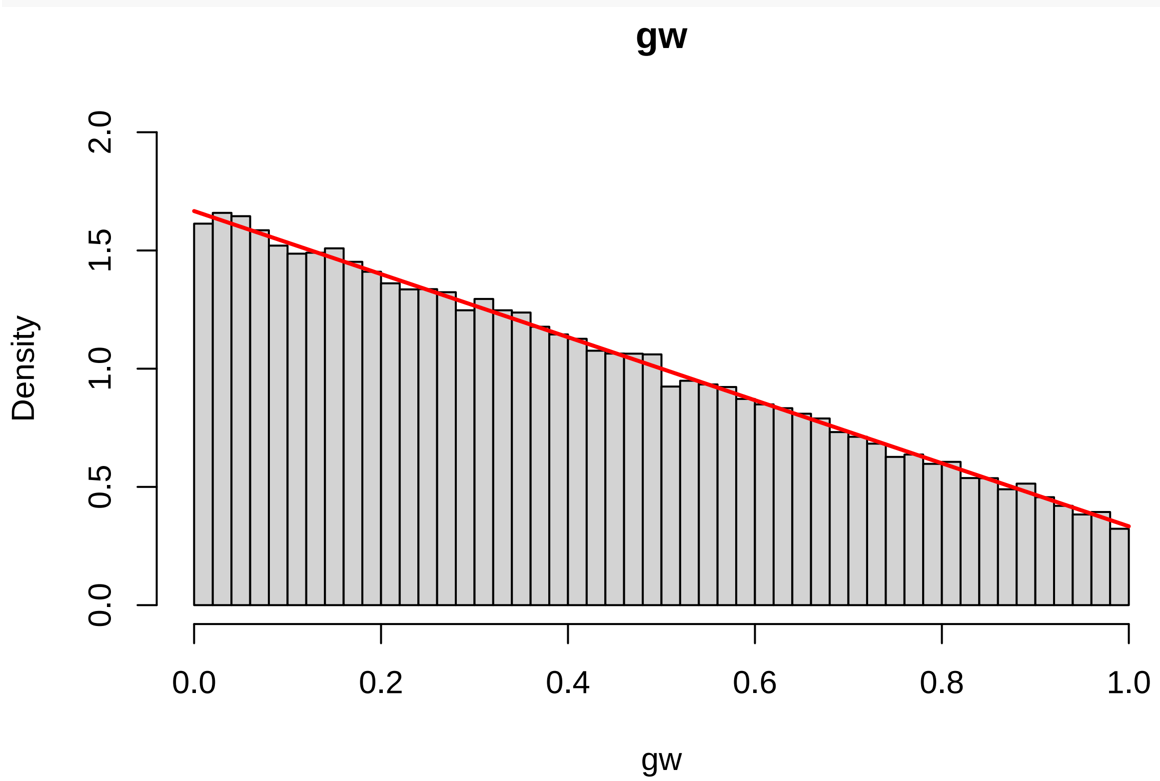

```r
hist(fz, breaks = 50, probability = TRUE, main = "fz", ylim = c(0, 2), xlim = c(0, 1))
lines(x_vals, f_density, col = "red", lwd = 2)
```



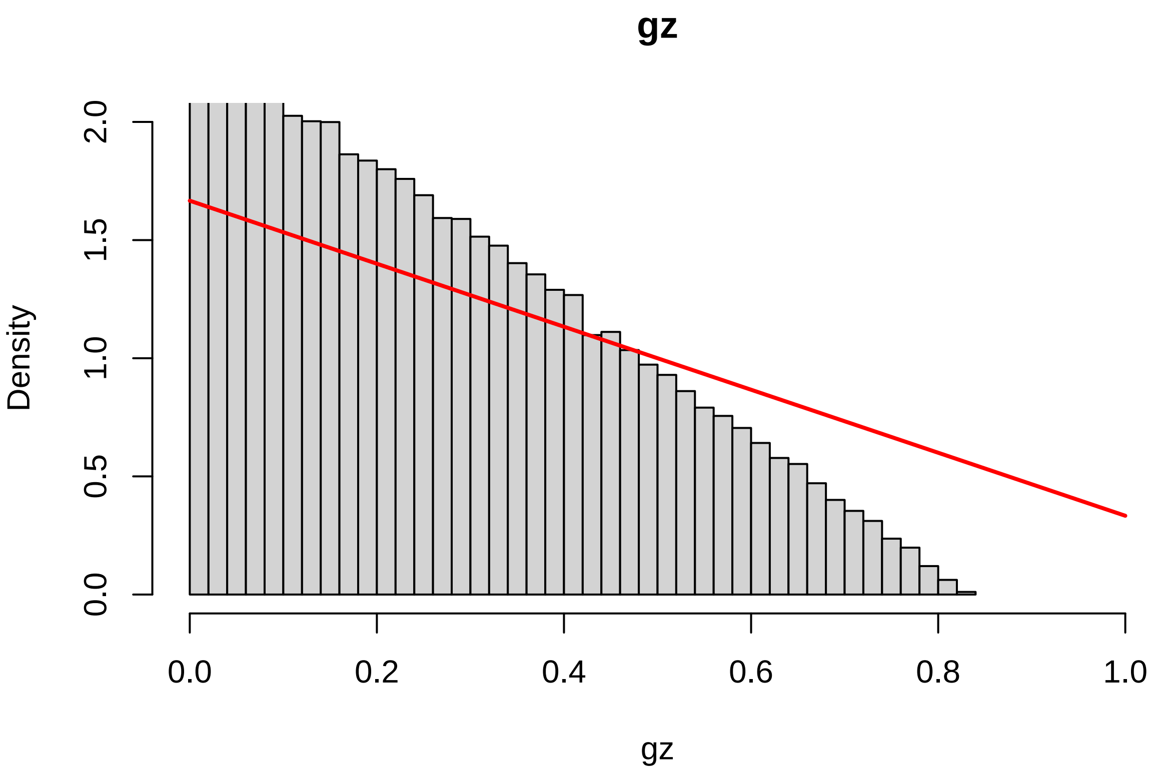

```r
hist(gv, breaks = 50, probability = TRUE, main = "gv", ylim = c(0, 2), xlim = c(0, 1))
lines(x_vals, f_density, col = "red", lwd = 2)
```




```r
hist(gw, breaks = 50, probability = TRUE, main = "gw", ylim = c(0, 2), xlim = c(0, 1))
lines(x_vals, f_density, col = "red", lwd = 2)
```




```r
hist(gz, breaks = 50, probability = TRUE, main = "gz", ylim = c(0, 2), xlim = c(0, 1))
lines(x_vals, f_density, col = "red", lwd = 2)
```



Based on our results above, we can see that our fv, gv, and gw are correct, as they correctly align with the true density $f(x)$. That is: $f(u_i)$ with v_i, $g(u_i)$ with v_i, and $g(u_i)$ with w_i.

2.

a)

MLE Estimates involve taking the derivative of the log-likelihood function and setting it to zero, and so:

$$\begin{aligned}\ln(L(y;\lambda,\kappa)) &= \sum_{i=1}^{n} \ln\left[\frac{\kappa}{\lambda}\left(\frac{y_i}{\lambda}\right)^{\kappa-1} e^{-(y_i/\lambda)^{\kappa}}\right] \\ &= \sum_{i=1}^{n} \ln[\kappa] - \ln[\lambda] + (\kappa-1)(\ln[y_i] - \ln[\lambda]) - (y_i/\lambda)^{\kappa}\end{aligned}$$

When taking the derivative and setting it to zero for λ, we get that the following equation must be satisfied:

$$0 = \sum_{i=1}^{n} -\frac{1}{\lambda} - \frac{\kappa-1}{\lambda} + \frac{\kappa y_i^{\kappa}}{\lambda^{\kappa-1}}$$

When taking the derivative and setting it to zero for κ, we get that the following equation must be satisfied:

$$0 = \sum_{i=1}^{n}(1/\kappa) - \left(\frac{y_i}{\lambda}\right)^{\kappa} \ln(y_i/\lambda) + \ln(y_i) - \ln(\lambda)$$

b)

$$\lambda_{MLE} = \left(\frac{1}{n}\sum_{i=1}^{n} y_i^{\kappa}\right)^{1/\kappa}$$

However, we cannot find a closed form solution for κ.

c)

We implement gradient descent.

```
set.seed(personal_number)

grad_kappa <- function(kappa, y) {
  n <- length(y)
  lambda <- (sum(y^kappa) / n)^(1/kappa)

  grad <- sum(-1/lambda - (kappa - 1)/lambda + (kappa * y^kappa) / lambda^(kappa - 1))

  return(grad)
}

estimate_kappa <- function(y, alpha = 0.001, tol = 1e-6, max_iter = 1000) {
  kappa <- 1
  for (i in 1:max_iter) {
    grad <- grad_kappa(kappa, y)
    kappa_new <- kappa - alpha * grad

    if (abs(kappa_new - kappa) < tol) {
      break
    }
    kappa <- kappa_new
  }
  return(kappa)
}
```

d)

```
set.seed(personal_number)
y <- rweibull(100, shape = 2, scale = 3)
kappa_hat <- estimate_kappa(y)
kappa_hat
```

```
## [1] 5.05801e-06
```

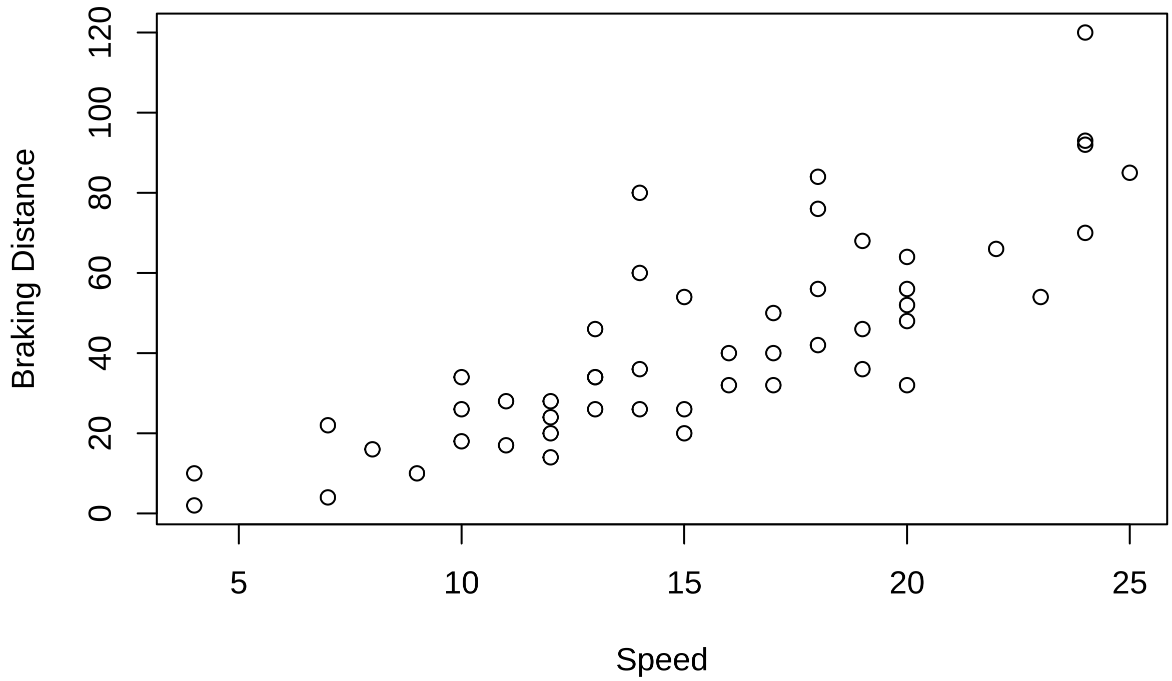
```
set.seed(personal_number)
y <- rweibull(1000, shape = 2, scale = 3)
kappa_hat <- estimate_kappa(y)
kappa_hat
```

```
## [1] -4.781885e-07
```

3.

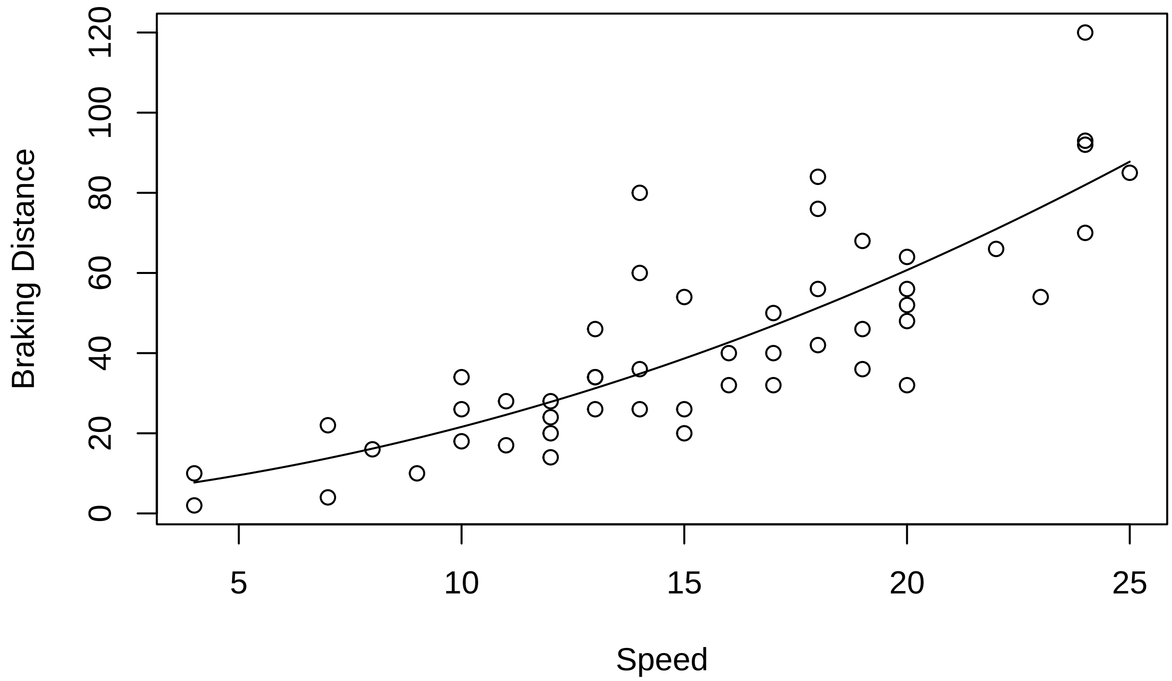
a)

```
cars_data <- cars
plot(cars_data$speed, cars_data$dist, xlab = "Speed", ylab = "Braking Distance")
```



b)

```
plot(cars_data$speed, cars_data$dist, xlab = "Speed", ylab = "Braking Distance")
X <- cbind(1, cars_data$speed, cars_data$speed^2)
Y <- cars_data$dist

coeffs <- solve(t(X) %*% X, t(X) %*% Y)
a <- coeffs[1]
b <- coeffs[2]
c <- coeffs[3]

speed_seq <- seq(min(cars_data$speed), max(cars_data$speed), length.out = 100)
dist_fit <- a + b * speed_seq + c * speed_seq^2
lines(speed_seq, dist_fit)
```

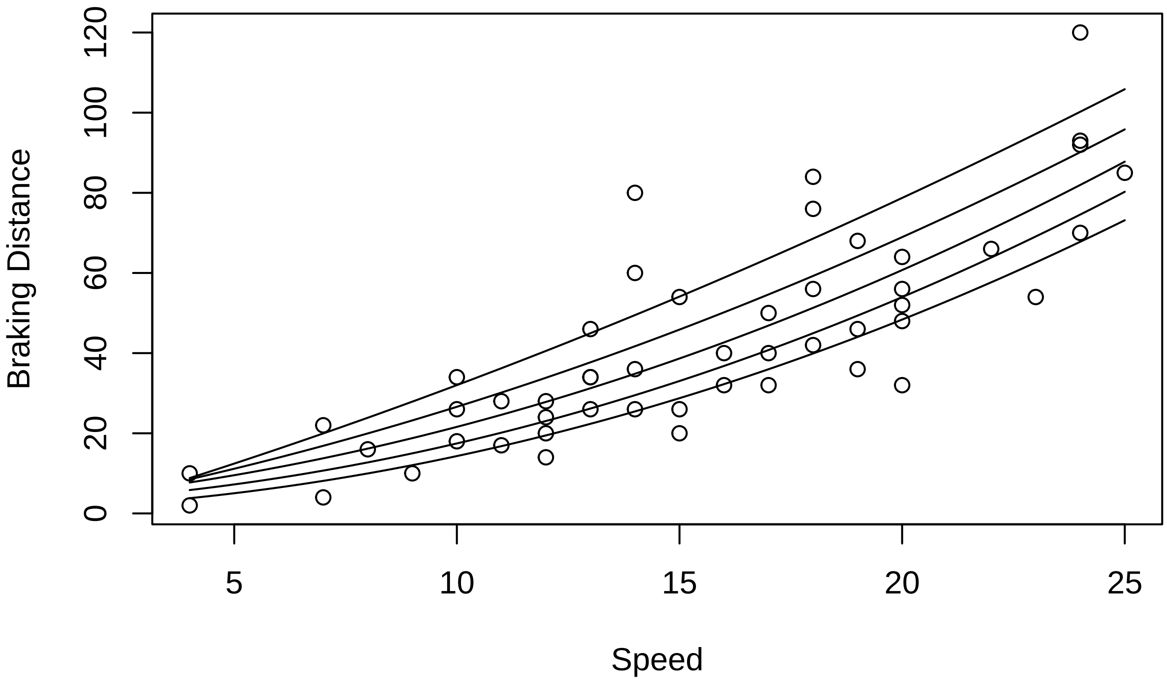


c)

```
quantile_regression <- function(x, y, p, tol = 1e-6, max_iter = 100) {
  n <- length(y)
  X <- cbind(1, x, x^2)
  beta <- solve(t(X) %*% X, t(X) %*% y)

  for (iter in 1:max_iter) {
    residuals <- y - X %*% beta
    weights <- p * (residuals > 0) + (1 - p) * (residuals < 0)
    W <- diag(as.vector(weights))
    new_beta <- solve(t(X) %*% W %*% X, t(X) %*% W %*% y)

    if (sum(abs(new_beta - beta)) < tol) break
    beta <- new_beta
  }
  return(beta)
}
```

d)

```
quantiles <- c(0.1, 0.25, 0.5, 0.75, 0.9)
plot(cars_data$speed, cars_data$dist, xlab = "Speed", ylab = "Braking Distance")
for (i in 1:length(quantiles)) {
  beta_q <- quantile_regression(cars_data$speed, cars_data$dist, quantiles[i])
  dist_q_fit <- beta_q[1] + beta_q[2] * speed_seq + beta_q[3] * speed_seq^2
  lines(speed_seq, dist_q_fit)
}
```



4.

a)

Rewriting our second provided inequality, we get $u \geq -v$ or $-v \leq u$. Then, $-v \leq u \leq v$, and so $|u| < v$