```r
# Generate sample data
set.seed(123)
n <- 100
x <- rnorm(n, mean = 5, sd = 2)

# Log-likelihood function for N(mu, sigma^2)
loglik <- function(params, x) {
  mu <- params[1]
  sigma2 <- params[2]
  n <- length(x)
  ll <- -n/2 * log(2 * pi * sigma2) - (1/(2 * sigma2)) * sum((x - mu)^2)
  return(ll)
}

# Score function (gradient)
score <- function(params, x) {
  mu <- params[1]
  sigma2 <- params[2]
  n <- length(x)
  dmu <- sum(x - mu) / sigma2
  dsigma2 <- -n/(2 * sigma2) + sum((x - mu)^2) / (2 * sigma2^2)
  return(c(dmu, dsigma2))
}

# Hessian (second derivatives)
hessian <- function(params, x) {
  mu <- params[1]
  sigma2 <- params[2]
  n <- length(x)
  d2mu2 <- -n / sigma2
  d2mu_sigma2 <- -sum(x - mu) / sigma2^2
  d2sigma2_2 <- n/(2 * sigma2^2) - sum((x - mu)^2) / (sigma2^3)
  H <- matrix(c(d2mu2, d2mu_sigma2, d2mu_sigma2, d2sigma2_2), nrow = 2)
  return(H)
}

# Newton-Raphson algorithm
newton_raphson <- function(x, init, tol = 1e-6, max_iter = 100) {
  params <- init
  for (i in 1:max_iter) {
    grad <- score(params, x)
    H <- hessian(params, x)
    step <- solve(H, grad)
    params_new <- params - step
    if (sum(abs(params_new - params)) < tol) {
      cat("Converged in", i, "iterations\n")
      break
    }
    params <- params_new
  }
  return(params)
}

# Initial guesses (sample mean and variance)
init <- c(mean(x), var(x))

# Run Newton-Raphson
mle_nr <- newton_raphson(x, init)
cat("Newton-Raphson MLEs:\nmu =", mle_nr[1], ", sigma^2 =", mle_nr[2], "\n")
```

```
# Closed-form MLEs
mu_mle <- mean(x)
sigma2_mle <- mean((x - mu_mle)^2)
cat("Closed-form MLEs:\nmu =", mu_mle, ", sigma^2 =", sigma2_mle, "\n")
```