```
## Warning: package 'knitr' was built under R version 4.4.2
```

2. Metropolis Random Walk for Poisson regression

We consider the Poisson regression model:

$$y_i|\beta \sim Poisson[exp(x_i^T\beta)], \quad i = 1, ..., n$$

To obtain the MLE of β, we fit the model using the `glm()` function with a Poisson likelihood.

```
# read data
data <- read.table("eBayNumberOfBidderData_2025.dat", header = T)

# (a) Poisson MLE using glm
mod <- glm(nBids ~ PowerSeller + VerifyID + Sealed + Minblem + MajBlem +
             LargNeg + LogBook + MinBidShare, data = data, family = "poisson")
coefs <- summary(mod)$coefficients
kable(coefs, caption = "Summary of the Poisson regression model")
```

Table 1: Summary of the Poisson regression model

	Estimate	Std. Error	z value	Pr(>|z|)
(Intercept)	1.0853366	0.0359229	30.2129332	0.0000000
PowerSeller	-0.0283250	0.0443252	-0.6390282	0.5228046
VerifyID	-0.3490151	0.1300773	-2.6831361	0.0072935
Sealed	0.5010095	0.0667643	7.5041528	0.0000000
Minblem	-0.1218935	0.0838830	-1.4531365	0.1461859
MajBlem	-0.2488434	0.0986451	-2.5226133	0.0116486
LargNeg	0.0361005	0.0707495	0.5102585	0.6098704
LogBook	-0.0727990	0.0350548	-2.0767205	0.0378274
MinBidShare	-1.7666504	0.0829198	-21.3055341	0.0000000

From this table, we observe that the covariates `VerifyID`, `Sealed`, `MajBlem`, `LogBook` and `MinBidShare` are statistically significant at the 5% level. This suggests that these variables have a meaningful effect on the expected number of bids in an eBay auction.

We now adopt a Bayesian approach to inference in the Poisson regression model. Specifically, we place a Zellner's g-prior on the regression coefficients β, defined as:

$$\beta \sim N[0, 100 \cdot (X^T X)^{-1}]$$

To approximate the posterior, we assume it is approximately multivariate normal:

$$\beta|y \sim N(\tilde{\beta}, J_Y^{-1}(\tilde{\beta}))$$

where $\tilde{\beta}$ is the posterior mode and $J_Y^{-1}(\tilde{\beta})$ is the negative Hessian and the posterior mode. We look for these values by numerical optimization. We obtain both via numerical optimization of the log-posterior using the BFGS algorithm:

```r
# (b)
y <- data$nBids
X <- model.matrix(mod)

# log-posterior function
log_posterior <- function(beta, X, y){
  XtX <- t(X) %*% X
  sigma0_inv <- (1/100) * XtX
  # llik
  eta <- X %*% beta
  llik <- sum(y * eta - exp(eta))

  # log-prior (gaussian with mean 0)
  logprior <- -0.5 * t(beta) %*% sigma0_inv %*% beta

  # Return scalar value
  log_post <- llik + logprior

  return(as.numeric(log_post))
}

# find posterior mode
# Initialize beta
init_beta <- rep(0, ncol(X))

# Optimization
opt_result <- optim(
  par = init_beta,
  fn = function(beta) -log_posterior(beta, X, y), # negative because optim minimizes
  method = "BFGS",
  hessian = TRUE
)

# Posterior mode and Hessian
beta_tilde <- opt_result$par
Hessian <- opt_result$hessian
```

Below are the estimated values of the posterior mode $\tilde{\beta}$

```r
kable(round(beta_tilde, 2), caption = "Posterior mode estimates of beta")
```

Table 2: Posterior mode estimates of beta

x
1.08
-0.03
-0.35
0.50
-0.12
-0.25
0.04
-0.07

x
-1.76

We also extract the observed information matrix $J_Y(\tilde{\beta})$

```
kable(round(Hessian, 2), caption = "Negative Hessian (observed information) at the posterior mode")
```

Table 3: Negative Hessian (observed information) at the posterior mode

2483.52	1050.55	63.15	271.74	154.05	109.95	241.82	408.67	-473.48
1050.55	1050.55	32.86	159.59	58.61	56.62	31.59	-98.47	-77.05
63.15	32.86	63.15	29.05	1.84	0.00	0.00	18.57	-10.46
271.74	159.59	29.05	271.74	0.00	0.00	0.00	19.20	-25.41
154.05	58.61	1.84	0.00	154.05	0.00	16.20	34.36	-24.07
109.95	56.62	0.00	0.00	0.00	109.95	0.00	27.41	-28.23
241.82	31.59	0.00	0.00	16.20	0.00	241.82	158.42	-77.24
408.67	-98.47	18.57	19.20	34.36	27.41	158.42	1301.18	-362.91
-473.48	-77.05	-10.46	-25.41	-24.07	-28.23	-77.24	-362.91	310.73

To obtain samples from the true posterior distribution of the regression coefficients β, we implement the Random Walk Metropolis-Hastings algorithm. The proposal distribution is defined as a multivariate normal random walk:

$$\theta_p|\theta^{(i-1)} \sim N(\theta^{(i-1)}, c \cdot \Sigma)$$

where $\Sigma = J_y^{-1}(\tilde{\beta})$.

The following code defines and runs the Metropolis sampler:

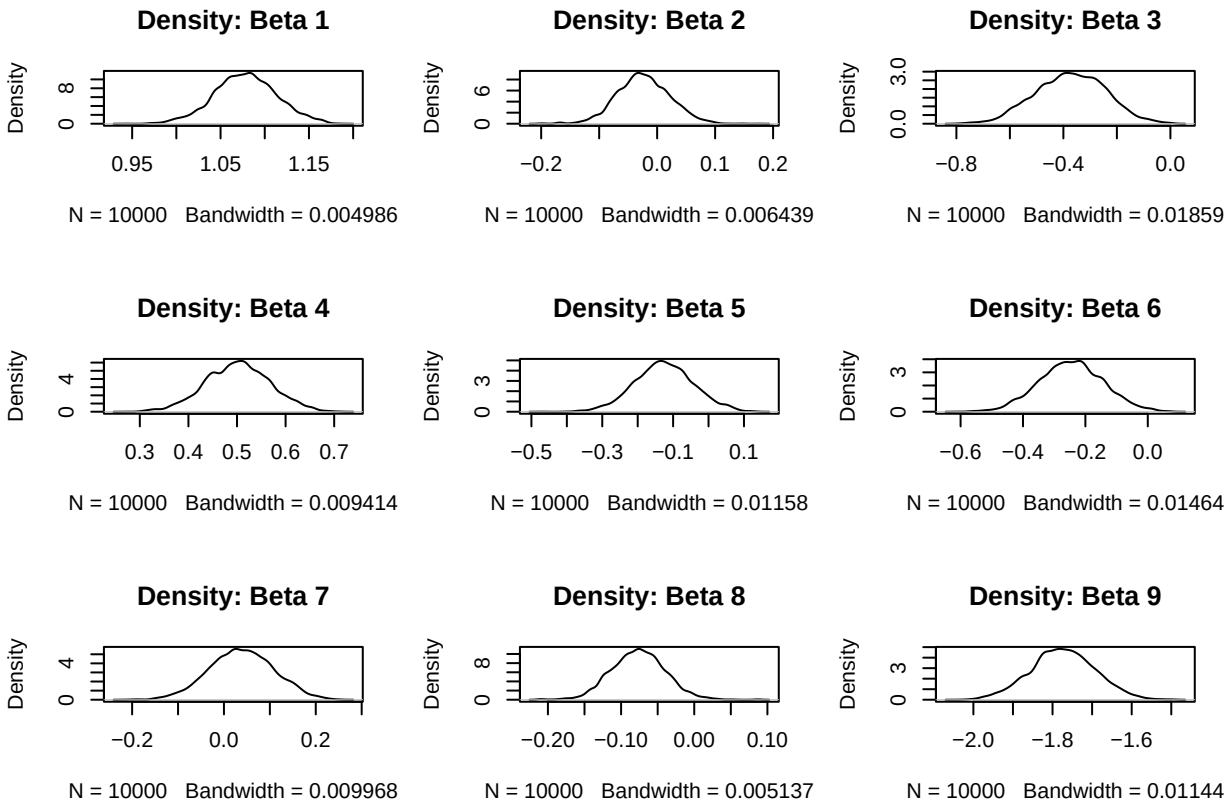
```
# (c)
MetropolisHastings <- function(x_init, n_iter, prop_dist_rand, target, X, y, Sigma, c){
  set.seed(1234)

  # initial values
  p <- length(x_init)
  x <- matrix(NA, nrow = n_iter, ncol = p)
  x[1, ] <- x_init
  acc <- 0

  for (t in 2:n_iter) {
    x_candidate <- prop_rand(x[t - 1, ], Sigma, c)
    x_candidate <- as.vector(x_candidate)

    log_r <- target(x_candidate, X, y) - target(x[t - 1, ], X, y)
    if (log(runif(1)) < log_r) {
      x[t, ] <- x_candidate
      acc <- acc + 1
    } else {
      x[t, ] <- x[t - 1, ]
    }
```

```
  }
  return(list(chain = x, acceptance_rate = sum(acc)/(n_iter-1)))
}

prop_rand <- function(theta, Sigma, c){
  mvtnorm::rmvnorm(1, mean = theta, sigma = c * Sigma)
}

#init_beta <- rep(0, ncol(X))  # or use beta_tilde from earlier optimization
Sigma <- solve(Hessian)

result <- MetropolisHastings(
  x_init = beta_tilde, # init_beta,
  n_iter = 10000,
  prop_dist = prop_rand,
  target = log_posterior,
  X = X,
  y = y,
  Sigma = Sigma,
  c = 0.5  # tuning parameter
)

posterior_samples <- result$chain
acceptance_rate <- result$acceptance_rate
```

We chose a tuning parameter `c` that yields an acceptance rate around 25–30% because this range is widely recommended for Random Walk Metropolis algorithms, balancing the trade-off between proposal acceptance and efficient exploration of the posterior distribution. We got an acceptance rate of 0.320332, which is acceptable.

To further explore the posterior distributions of the parameters β_j we plot the estimated marginal densities

```
# Density plots
par(mfrow = c(3, 3))
for (j in 1:ncol(posterior_samples)) {
  plot(density(posterior_samples[, j]), main = paste("Density: Beta", j))
}
```



Using the MCMC samples, we now simulate from the posterior predictive distribution of the number of bidders in a new auction with the following characteristics:

- PowerSeller = 1
- VerifyID = 0
- Sealed = 1
- MinBlem = 0
- MajBlem = 1
- LargNeg = 0
- LogBook = 1.3
- MinBidShare = 0.7

This involves two steps:

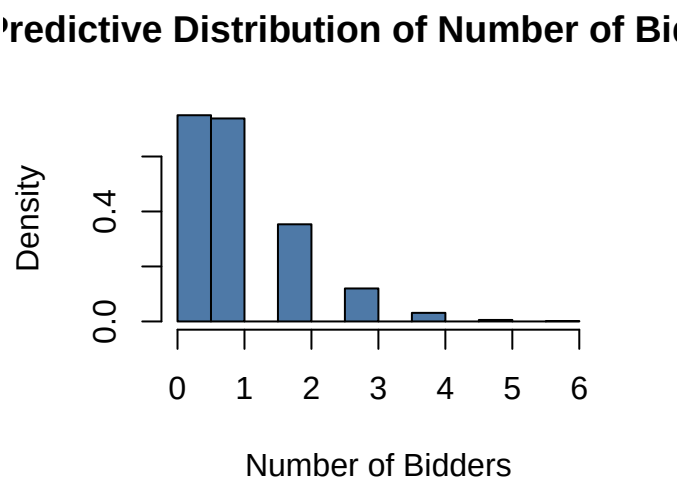
1. Compute the rate parameter λ for each posterior sample of β:

$$\lambda^{(s)} = exp(x_{new}^T \beta^{(s)}), \quad s = 1, ..., S$$

2. Simulate the number of bidders from a Poisson distribution using each $\lambda^{(s)}$

$$\tilde{y}^{(s)} \sim Poisson(\lambda^{(s)})$$

```r
# (d)
x_new <- c(1,    # Intercept
           1,    # PowerSeller
           0,    # VerifyID
           1,    # Sealed
           0,    # MinBlem
           1,    # MajBlem
           0,    # LargNeg
           1.3,  # LogBook
           0.7)  # MinBidShare

# Compute lambda for each posterior draw
lambda_draws <- apply(posterior_samples, 1, function(beta) {
  exp(sum(beta * x_new))
})

# Simulate nBids for new auction
set.seed(1234)
pred_nBids <- rpois(length(lambda_draws), lambda = lambda_draws)

# Plot predictive distribution
par(mfrow = c(1,1))
hist(pred_nBids, col = "#4E79A7", probability = TRUE, breaks = 20,
     main = "Predictive Distribution of Number of Bidders",
     xlab = "Number of Bidders")
```




```r
# Estimate P(nBids = 0)
prob_zero_bidders <- mean(pred_nBids == 0)
```

The probability of no bidders in this new auction is 0.3751.