a1

2024-09-12

Question 1).

```r
unirand <- function(n, m=30269, a=171, seed=1) {
  x <- numeric(min(m-1,n))
  x[1] <- seed
  for (i in 1:min(m-1,n)){
    y <- x[i]
    x[i+1] <- (a*y)%%m
  }
  x[2:(n+1)]/m
}
unirand(5)
```

```
## [1] 0.005649344 0.966037861 0.192474148 0.913079388 0.136575374
```

```r
# Code taken from SimulationI slide
```

This multiplicative congruential generator contains a maximal cycle length of $m - 1 = 30306$.

Question 2).

```r
unirand2 <- function(n, m=30323, a=170, seed=1) {
  x <- numeric(min(m-1,n))
  x[1] <- seed
  for (i in 1:min(m-1,n)){
    y <- x[i]
    x[i+1] <- (a*y)%%m
  }
  x[2:(n+1)]/m
}
unirand2(5)
```

```
## [1] 0.005606305 0.953071926 0.022227352 0.778649870 0.370477855
```

```r
# Code taken from SimulationI slide
```

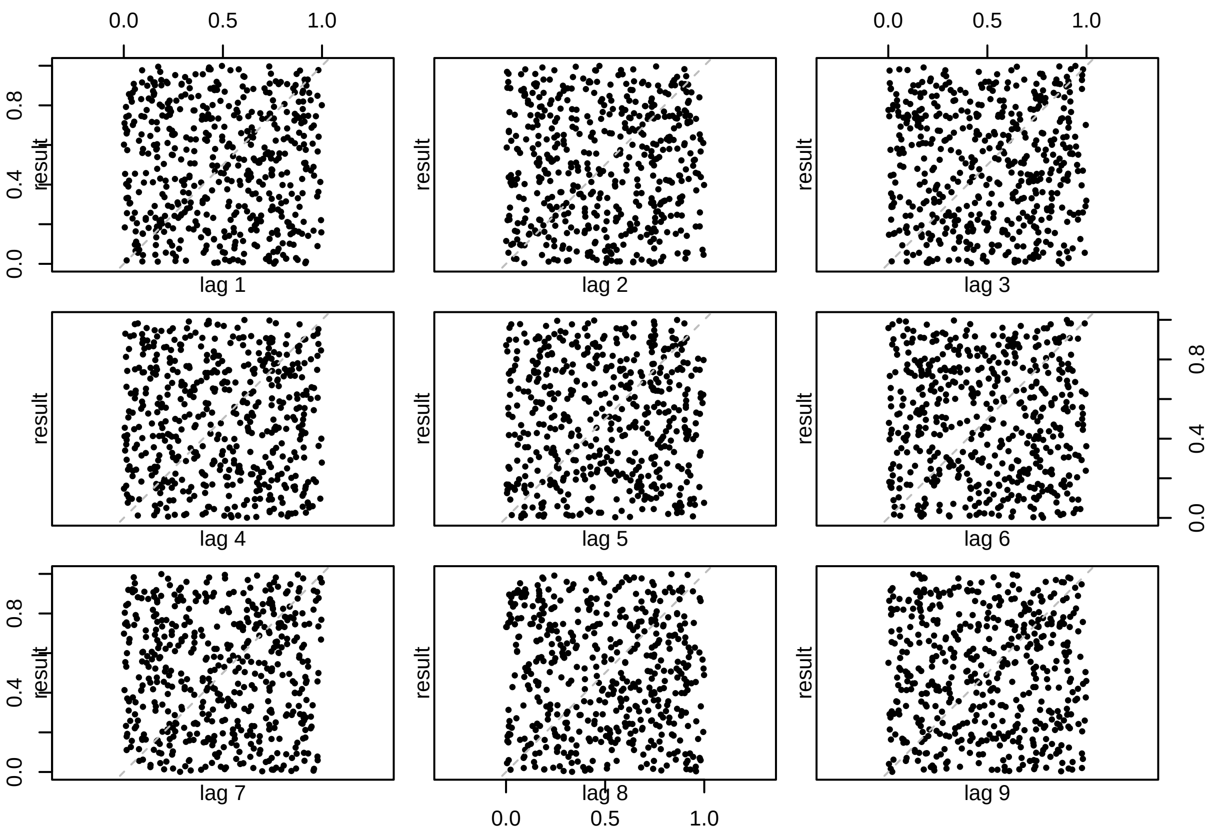
This multiplicative congruential generator contains a maximal cycle length of $m - 1 = 30322$.

Question 3).

```r
unirand3 <- function(n, s1, s2) {
  u1 <- unirand(n, seed=s1)
  u2 <- unirand2(n, seed=s2)
  u3 <- u1 + u2 - floor(u1 + u2)
  return(u3)

}

result <- unirand3(500, 1, 1)
lag.plot(result, lag=9, pch=16, do.lines=FALSE)
```



By these lag graphs, we can tell that the plots are uniformly distributed, and they do not seem to be dependent of the past plots. The cycle length of this generator cannot exceed 300000, because the m of unirand and unirand2 are both below 300000, where the cycle length is determined by taking the minimum of $m - 1$ and n where $n = 300000$.

Question 4).

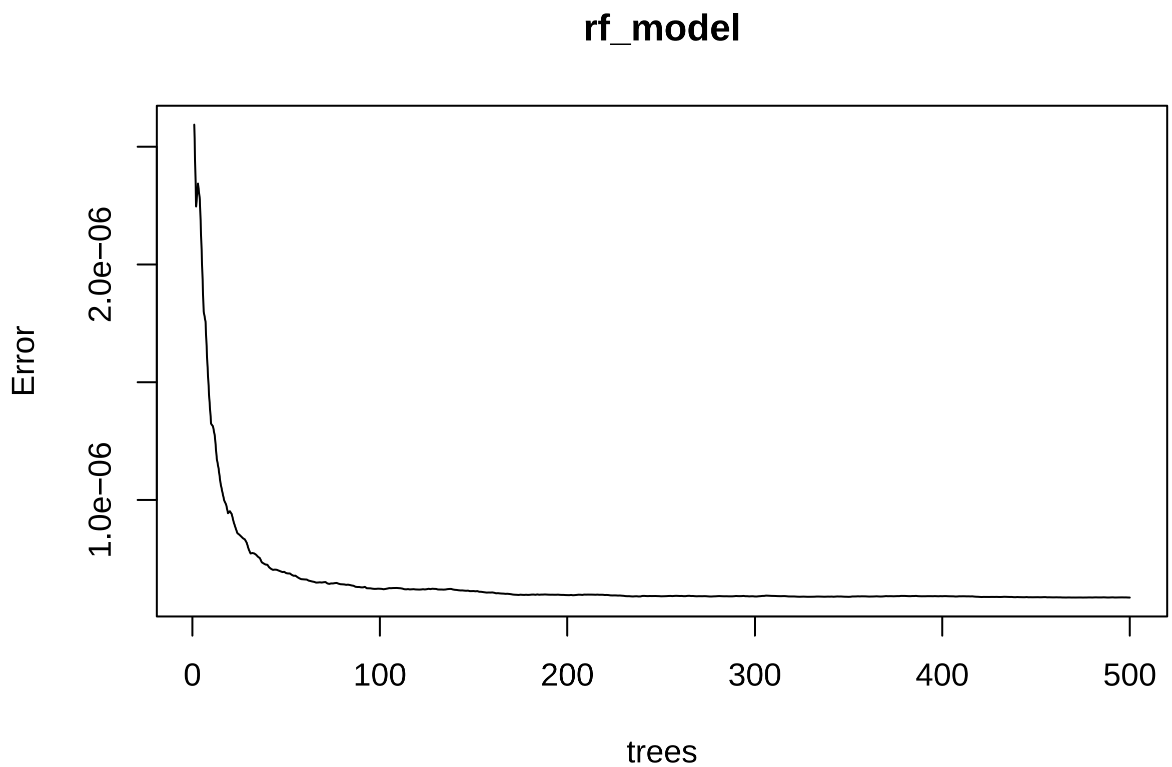
```r
result1000 <- unirand3(1000, 1, 1)
df <- data.frame(y = result1000, x1 = result1000, x2 = result1000, x3 = result1000,
                 x4 = result1000, x5 = result1000, x6 = result1000, x7 = result1000,
```

```r
                        x8 = result1000, x9 = result1000, x10 = result1000)
library(randomForest)
```

```
## randomForest 4.7-1.1

## Type rfNews() to see new features/changes/bug fixes.
```

```r
rf_model <- randomForest(y~.,data = df)
plot(rf_model)
```




```r
print(rf_model)
```

```
##
## Call:
##  randomForest(formula = y ~ ., data = df)
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 3
##
##           Mean of squared residuals: 5.856468e-07
##                     % Var explained: 100
```

With the plot and the printed information, we can safely say that the randomForest follows a very high percentage of variance at 100%, which makes the model overfitting, that means the model is allowing less error as more trees are generated.

Question 5

```r
# find greatest common denominator
gcd <- function(a, b) {
  while (b != 0) {
    temp <- b
    b <- a %% b
    a <- temp
  }
  return(a)
}

# helper function
gcd_set <- function(numbers) {
  result <- numbers[1]
  for (i in 2:length(numbers)) {
    result <- gcd(result, numbers[i])
  }
  return(result)
}

# example
numbers <- c(2205,21065,5241, 12752, 25817,6724,18604,7158,21788,20601)
gcd_result <- gcd_set(numbers)
print(gcd_result)
```

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