Data 405 Assignment 1 2025W

2025-9-25

Data 405 Assignment 1

Q1 code:

```
unirand <- function(n, s) {
    # moduluo & multiplier variables
    m <- 30307
    a <- 172

    x <- integer(n)   # raw sequence
    u <- numeric(n)   # normalized seq

    x[1] <- (a * s) %% m
    u[1] <- x[1] / m

    if (n > 1) {
        for (i in 2:n) {
            x[i] <- (a * x[i-1]) %% m
            u[i] <- x[i] / m
        }
    }
    return(u)
}

# s is for seed, n is for number, m is for modulo, a is for multiplier, x is for raw sequence, u is for
```

```
# 10 uniforms using unirand1 with seed = 13
unirand(10, 13)
```

```
##  [1] 0.07377834 0.68987363 0.65826377 0.22136800 0.07529614 0.95093543
##  [7] 0.56089352 0.47368595 0.47398291 0.52506022
```

Q1 discussion:

The maximal cycle length of a multiplicative congruential generator with moduluo m requires that the 'a' variable be a primitive root modulo m. Since 172 is NOT a primitive root of modulo 30307, this generator does NOT achieve maximal cycle length (which would be m-1 = 30306)

Q2 code:

```
unirand2 <- function(n, s) {
    m <- 30323
    a <- 170

    x <- integer(n)
    u <- numeric(n)

    x[1] <- (a * s) %% m
    u[1] <- x[1] / m

    if (n > 1) {
        for (i in 2:n) {
            x[i] <- (a * x[i-1]) %% m
            u[i] <- x[i] / m
        }
    }
    return(u)
}

# s is for seed, n is for number, m is for modulo, a is for multiplier, x is for raw sequence, u is for
```

```
# 10 uniforms using unirand2 with seed = 13
unirand2(10, 42)
```

```
##  [1] 0.23546483 0.02902088 0.93354879 0.70329453 0.56006991 0.21188537
##  [7] 0.02051248 0.48712199 0.81073772 0.82541305
```

Q2 discussion:

For maximal cycle length: 'a' must be a primitive root modulo 'm', & since 170 is not a primitive root modulo on 30323, this generator also does NOT achieve maximal cycle length

Q3 code:

```
unirand3 <- function(n, s1, s2) {
    u1 <- unirand(n, s1)
    u2 <- unirand2(n, s2)

    u <- u1 + u2 - floor(u1 + u2)  # I believe this is [0,1)
    return(u)
}
```

```
# 10 uniforms using unirand3 with seed = 67
unirand3(10, 13, 9162)
```

```
##  [1] 0.43874882 0.73485598 0.30526439 0.21147452 0.39340450 0.02935775
##  [7] 0.89268787 0.87872503 0.33062638 0.15445045
```

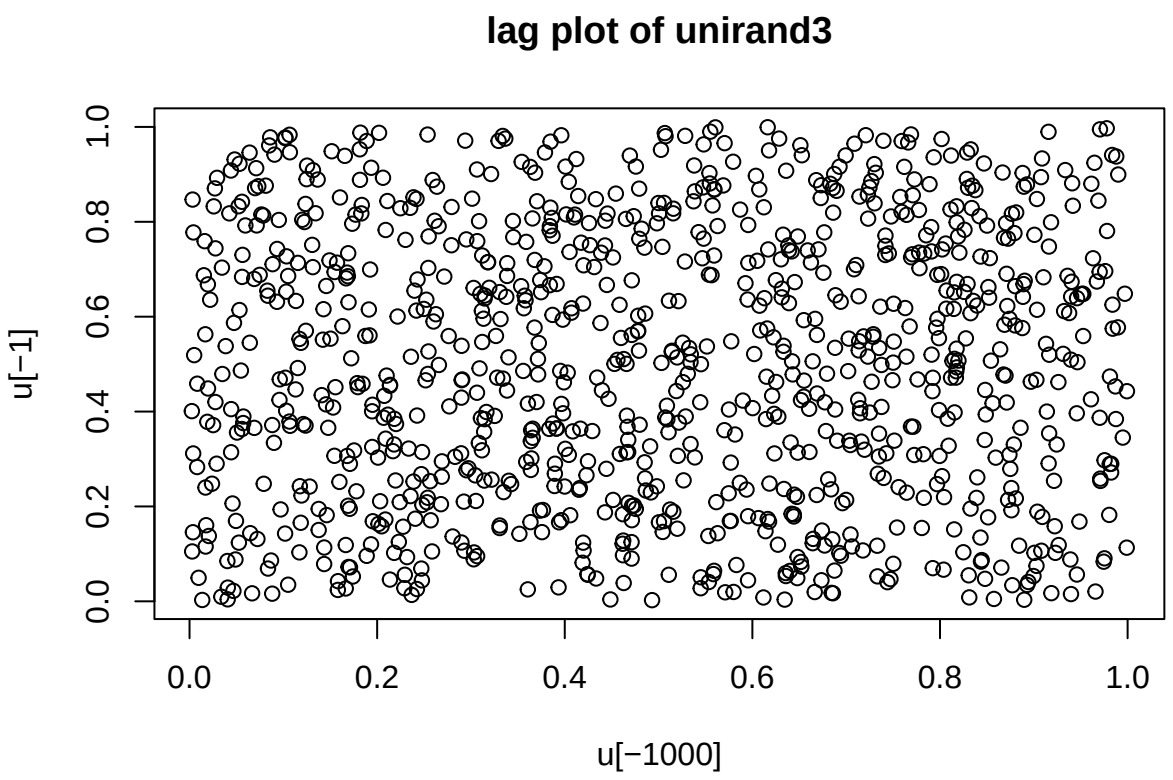
Q3 discussion:

When seeds are s1 = 13 and s2 = 9162, the cycle length exceeds 300000: - suppose m1-1 = 30306 and m2-1 = 30322 are the periods of the two individual generators, - the LCM will l be 459,469,266 (source https://www.calculatorsoup.com/calculators/math/lcm.php) - thus: cycle length »> 300,000

The resulting numbers look to be approximately uniformly distributed on [0,1)

To check independence, we can lag plot & use a linear model:

```
u <- unirand3(1000, 13, 9162)
plot(u[-1000], u[-1], main="lag plot of unirand3")
```



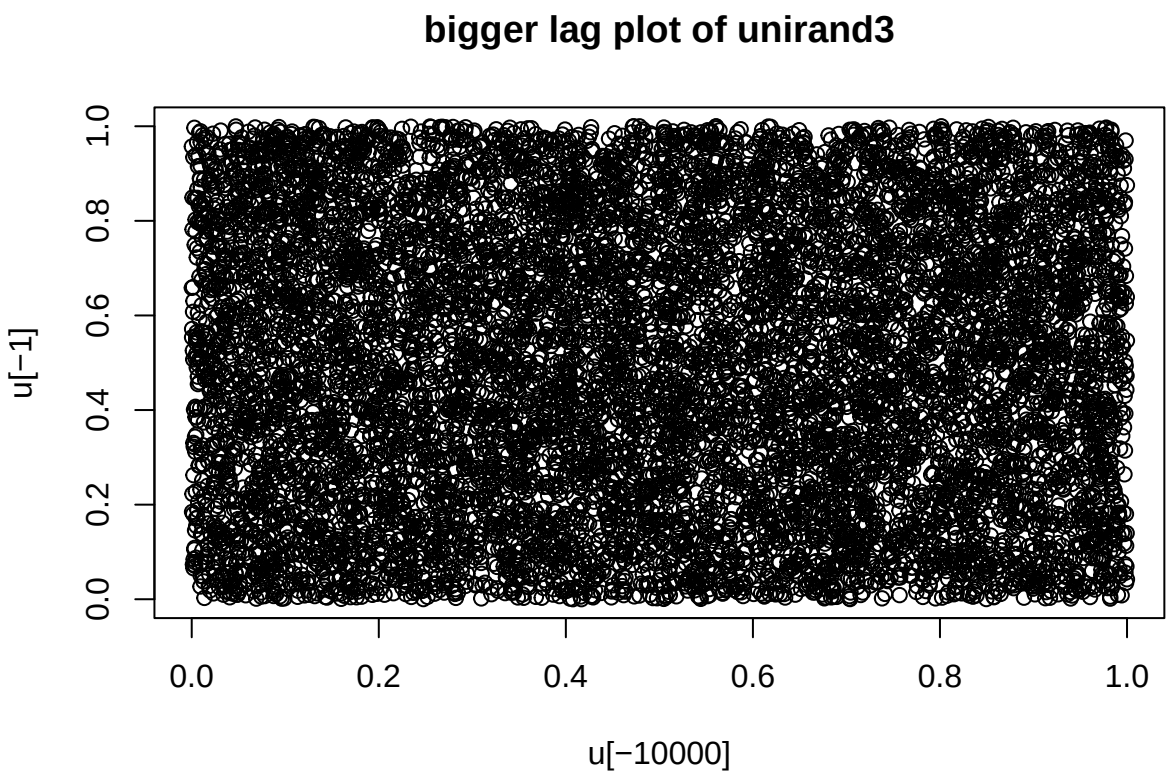

```
# lagged version
u_lag <- u[-length(u)]        # u[1 ... n-1]
u_now <- u[-1]                # u[2 ..... n]

# lin reg
fit <- lm(u_now ~ u_lag)

summary(fit)
```

```
##
## Call:
## lm(formula = u_now ~ u_lag)
##
```

```
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.50853 -0.24444 -0.00065  0.23966  0.49928
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.48264    0.01815  26.593   <2e-16 ***
## u_lag        0.03281    0.03165   1.037      0.3
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.2827 on 997 degrees of freedom
## Multiple R-squared:  0.001077,   Adjusted R-squared:  7.489e-05
## F-statistic: 1.075 on 1 and 997 DF,  p-value: 0.3001
```

an intercept close to 0.5 & a slope close to 0 indicates uniform distributed data (0.48 & 0.03 respectively) but just to be sure, lets up sample from 1000 to 10,000 and check again:

```
u <- unirand3(10000, 13, 9162)
plot(u[-10000], u[-1], main="bigger lag plot of unirand3")
```




```
# lagged version
u_lag <- u[-length(u)]        # u[1 ... n-1]
u_now <- u[-1]                # u[2 ..... n]
```

```r
# lin reg
fit <- lm(u_now ~ u_lag)

summary(fit)
```

```
##
## Call:
## lm(formula = u_now ~ u_lag)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.49811 -0.25232 -0.00252  0.24960  0.50516
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.499862   0.005750  86.935   <2e-16 ***
## u_lag       -0.006374   0.010002  -0.637    0.524
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.2895 on 9997 degrees of freedom
## Multiple R-squared:  4.062e-05,  Adjusted R-squared:  -5.941e-05
## F-statistic: 0.4061 on 1 and 9997 DF,  p-value: 0.524
```

slope & intercept of 0.499 & -0.006 respectively show these numbers converge to 0.5 & 0 - indicating uniform distribution (as what little varrience from 0.5 & 0 is within acceptable noise levels, or simply a corner missing a data point)

Q4 code:

```r
set.seed(67)
u <- unirand3(1000, 13, 9162) # 1000 from uni rand 3

random_forest_test <- function(u, m, bins = 5) {
    n <- length(u) - m + 1
    tuples <- embed(u, m)  # form overlapping m-tuples on each
    # then we partition cube into bins^m cells
    cuts <- lapply(1:m, function(i) cut(tuples[,i], breaks=bins, labels=FALSE))
    idx <- do.call(paste, c(cuts, sep="-"))
    freq <- table(idx)

    # also a chi-square test for uniformity
    chisq.test(freq)
}

# dimension m=10 limit
results <- lapply(2:10, function(m) random_forest_test(u, m))
```

```
## Warning in chisq.test(freq): Chi-squared approximation may be incorrect
```

```
## Warning in chisq.test(freq): Chi-squared approximation may be incorrect

## Warning in chisq.test(freq): Chi-squared approximation may be incorrect

## Warning in chisq.test(freq): Chi-squared approximation may be incorrect

## Warning in chisq.test(freq): Chi-squared approximation may be incorrect

## Warning in chisq.test(freq): Chi-squared approximation may be incorrect

## Warning in chisq.test(freq): Chi-squared approximation may be incorrect
```

```
names(results) <- paste0("m=", 2:10)
results
```

```
## $`m=2`
##
##  Chi-squared test for given probabilities
##
## data:  freq
## X-squared = 26.951, df = 24, p-value = 0.3067
##
##
## $`m=3`
##
##  Chi-squared test for given probabilities
##
## data:  freq
## X-squared = 123.99, df = 124, p-value = 0.4833
##
##
## $`m=4`
##
##  Chi-squared test for given probabilities
##
## data:  freq
## X-squared = 302.56, df = 501, p-value = 1
##
##
## $`m=5`
##
##  Chi-squared test for given probabilities
##
## data:  freq
## X-squared = 128.8, df = 849, p-value = 1
##
##
## $`m=6`
##
##  Chi-squared test for given probabilities
##
## data:  freq
## X-squared = 30.131, df = 964, p-value = 1
##
```

```
##
## $`m=7`
##
##  Chi-squared test for given probabilities
##
## data:  freq
## X-squared = 6.9014, df = 986, p-value = 1
##
##
## $`m=8`
##
##  Chi-squared test for given probabilities
##
## data:  freq
## X-squared = 0.99799, df = 991, p-value = 1
##
##
## $`m=9`
##
##  Chi-squared test for given probabilities
##
## data:  freq
## X-squared = 0, df = 991, p-value = 1
##
##
## $`m=10`
##
##  Chi-squared test for given probabilities
##
## data:  freq
## X-squared = 0, df = 990, p-value = 1
```

Q4 discussion:

Results for small m (2–3):

m=2: p=0.3067 p=0.3067 — no evidence against uniformity.

m=3: p=0.4833 p=0.4833 — again, no problem.

m>=4: p=1 p=1 — the test is not valid here due to insufficient sample size for the number of bins

The test works fine for low dimensions (m=2,3) where the sample size is sufficient

for higher dimensions (m >= 4), with only 1000 samples, the test likely isn't valid