Analytics 511: Homework #1

Brody Vogel

September 15th

Preparation

```
set.seed(1234)
```

Problem 1

a = 9

```
options(digits = 12)
a <- as.numeric(substr(sin(1.23), 12, 12))
a
```

```
## [1] 9
```

$b \approx 28.46$

```
b <- sqrt(a^2 + a^3)
b
```

```
## [1] 28.4604989415
```

c = 17

```
c <- nchar(round(exp(log(b)^3), 0))
c
```

```
## [1] 17
```

d = 23409

```
d <- sum(sapply(1:c, function(x)x^3))
d
```

```
## [1] 23409
```

e = 1

```
e <- d%%77
e
```

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

Problem 2

```r
mytoss <- function(p) {
  u <- runif(1)
  x <- as.numeric(u < p)
  return(x)
}

myattempts <- function(p) {
  counter <- 1
  while (mytoss(p) == 0) {
    counter <- counter + 1
  }
  return(counter)
}

tester <- function(p) {
  time1 <- system.time(replicate(2000, myattempts(p)))
  time2 <- system.time(replicate(2000, rgeom(1, p)))
  print(p)
  print(time1)
  print(time2)
}
ps <- c(.01, .1, .4, .8, .99)
sapply(ps, function(x)(tester(x)))
```

```
## [1] 0.01
##    user  system elapsed
##   0.534   0.010   0.545
##    user  system elapsed
##   0.004   0.000   0.005
## [1] 0.1
##    user  system elapsed
##   0.047   0.000   0.047
##    user  system elapsed
##   0.003   0.000   0.004
## [1] 0.4
##    user  system elapsed
##   0.014   0.000   0.014
##    user  system elapsed
##   0.004   0.000   0.003
## [1] 0.8
##    user  system elapsed
##   0.008   0.000   0.008
##    user  system elapsed
##   0.003   0.000   0.003
## [1] 0.99
##    user  system elapsed
##   0.006   0.000   0.006
##    user  system elapsed
```

```
##   0.003   0.001   0.004
```

```
##             [,1]  [,2]  [,3]  [,4]  [,5]
## user.self  0.004 0.003 0.004 0.003 0.003
## sys.self   0.000 0.000 0.000 0.000 0.001
## elapsed    0.005 0.004 0.003 0.003 0.004
## user.child 0.000 0.000 0.000 0.000 0.000
## sys.child  0.000 0.000 0.000 0.000 0.000
```

The myattempts(p) function got significantly faster for larger values of p, since it didn't have to simulate as many trials. The built-in rgeom(1, p) function, though, stayed pretty consistent; consistently *faster*, to be specific. I'm not sure why this was the case, but I assume the built-in function pulls its values form some sort of massive table.

Problem 3

```
options(digits = 2)
tester1 <- function(p){
  simulated_sd <- sd(replicate(10000, rgeom(1, p)))
  actual_sd <- sqrt(1-p)/p
  sprintf("p: %f ; Simulated st.D: %f ; Actual st.D: %f", p, simulated_sd, actual_sd)
}
sapply(ps, function(x)tester1(x))
```

```
## [1] "p: 0.010000 ; Simulated st.D: 99.788674 ; Actual st.D: 99.498744"
## [2] "p: 0.100000 ; Simulated st.D: 9.677470 ; Actual st.D: 9.486833"
## [3] "p: 0.400000 ; Simulated st.D: 1.990757 ; Actual st.D: 1.936492"
## [4] "p: 0.800000 ; Simulated st.D: 0.542283 ; Actual st.D: 0.559017"
## [5] "p: 0.990000 ; Simulated st.D: 0.097513 ; Actual st.D: 0.101010"
```

There are no significant differences between the simulated standard deviations and the known standard deviations.

Problem 4

```
draws <- sapply(1:10, function(x)mean(rexp(20)))
draws
```

```
##  [1] 0.86 1.10 0.90 1.07 1.01 1.03 1.04 1.01 1.22 1.14
```

Problem 5

The strongest argument in support of R is that it's open source, i.e., it's free. R's being open source also creates a space for independent developers to share a huge number of packages for solving most conceivable problems a data scientist could encounter. Speaking just in terms of 'what it can do', then, R is the superior product.

SAS, it appears, is better than R at handling massive - like TBs - amounts of data. Without sufficient memory space, R cannot handle these kinds of datasets, while SAS can usually handle them in seconds. Also, since SAS isn't open source, it's a bit more secure than R.

All in all, though, it seems both R and SAS can do most anything a data scientist could need them to.

Problem 6

a) Pr(Not in Sample) = .00001

```
p <- 1000 / 100000000
p
```

```
## [1] 1e-05
```

b) Pr(Not in Any Sample) = Pr(Not in 1 Sample)$^{2000} \approx .98$

```
p2 <- (1-p)^2000
p2
```

```
## [1] 0.98
```

c) $.5 = (1 - p)^x$; $x \approx 69315$

```
num_til <- logb(.5, base = (1-p))
num_til
```

```
## [1] 69314
```

Problem 7

$bias(.1) \approx .006$; $bias(.3) \approx .052$; $bias(.7) \approx .378$

```
tester2 <- function(p) {
  x_bar <- mean(replicate(20000, mean(rgeom(4, p)^2)))
  estimator <- (sqrt(1 + 8 * x_bar) - 1)/(2 * x_bar)
  bias <- estimator - p
  sprintf("True p: %f ; Estimated p: %f ; bias: %f", p, estimator, bias)
}
ps1 <- c(.1, .3, .7)
sapply(ps1, function(x)(tester2(x)))
```

```
## [1] "True p: 0.100000 ; Estimated p: 0.104442 ; bias: 0.004442"
## [2] "True p: 0.300000 ; Estimated p: 0.352148 ; bias: 0.052148"
## [3] "True p: 0.700000 ; Estimated p: 1.074796 ; bias: 0.374796"
```

Problem 8

$E[X_1] = .5$ so $2E[X_1] = 1$. Thus $E[X_2] = .5$ and $2E[X_2] = 1$. So on and so forth, until $E[X_{10}] = .5$, which is mirrored in the simulation.

```
tester3 <- function() {
  start <- runif(1)
  for (x in 2:10) {
    start <- runif(1, min = 0, max = 2 * start)
  }
  return(start)
}
simulation <- mean(replicate(20000, tester3()))
simulation
```

```
## [1] 0.51
```