Appendix A: R code for generating Random Variates

R code for the algorithms defined by Kemp (1981) as well as the alias method preceded by a discussion of the essential functions follow.

*set.seed* – This sets R’s random number generator state so identical sets of variables could be reproduced.

*runif(n,a,b)* – This creates *n* randomly generated variables with a continuous uniform distribution from a to b.

*function(var1, var2, …, varN)* – This creates a user-defined function with *N* variables. Each algorithm was made into a function so that later studies could use different algorithms depending on the situation, i.e., if a logarithmic distribution did not have α = .15, use one of Kemp’s algorithms, else use the finitized alias method.

*Log(x)* – This takes the natural log of *x*.

*as.matrix(k)* – This creates a matrix object of list *k*.

*length(x)* – This counts the characters or units within *x*.

*ifelse(condition,1,0)* – This sets an object to 1 if *condition* is true, else 0.

*hist(list)* – This creates a simple histogram of *list*.

*summary(list)* – This gives summary statistics of *list.*

*write.table(table,file=…)* – This exports *table* to the file specified.

R Code written by Mr. Andy Kirtland (2009), modified from SAS code written by Dr. Martin S. Levy

#-----------------------------------------------------------------

#generates alias method logaritmic from finitized dist

ALIAS.Method <- function(reps,f,l,seed)

{

m <- length(f) - 1

t <- 1 / (m + 1)

set.seed(seed)

i <- as.integer(runif(reps,1,m+2))

ALIAS.Results <- 1 + ifelse( runif(reps,0,1) <= f[1,i], i-1, l[1,i])

as.matrix(ALIAS.Results)

}

#-----------------------------------------------------------------

#generates real logarithmic using Method LS (see Kemp, 1981)

LS.Method <- function(reps, a, seed)

{

t <- -a / (log(1-a))

set.seed(seed)

LS.Results <- NULL

for (j in 1:reps)

{

u1 <- runif(1,0,1)

x <- 1

p <- t

while (u1 > p)

{

u1 <- u1 - p

x <- x + 1

p = (p \* a \* (x-1)) / x

}

LS.Results[j] <- x

}

as.matrix(LS.Results)

}

#-----------------------------------------------------------------

#generates real logarithmic using Method LB (see Kemp, 1981)

LB.Method <- function(reps, a, seed)

{

h <- log(1 - a)

set.seed(seed)

LB.Results <- as.integer(1+log(runif(reps,0,1))/log(1-exp(h\*runif(reps,0,1))))

as.matrix(LB.Results)

}

#-----------------------------------------------------------------

#generates real logarithmic using Method LBM (see Kemp, 1981)

LBM.Method <- function(reps, a, seed)

{

h <- log(1 - a)

set.seed(seed+1)

u2 <- runif(reps,0,1)

LBM.Results <- ifelse( u2 > a, 1, as.integer(1+log(u2)/log(1-exp(h\*runif(reps,0,1)))))

as.matrix(LBM.Results)

}

#-----------------------------------------------------------------

#generates real logarithmic using Method LK (see Kemp, 1981)

LK.Method <- function(reps, a, seed)

{

h <- log(1-a)

set.seed(seed)

u2 <- runif(reps,0,1)

q <- 1 - exp(runif(reps,0,1)\*h)

LK.Results <- ifelse(u2 > q, 1, ifelse( u2 < q^2, as.integer(1 + log(u2) / log(q)), 2))

as.matrix(LK.Results)

}

#-----------------------------------------------------------------

#Frequency Procedure (***discrete case only***)

freq <- function(mat)

{

k <- 1

rows <- max(mat,na.rm = TRUE) + 1

freq.mat <- matrix(0,nrow=rows,ncol=3,dimnames = list(c(1:rows),c(‘Value’, ‘Count’, ‘Percent’)))

tot <- length(mat)

for (j in min(mat,na.rm = TRUE):max(mat,na.rm = TRUE))

{

k <- k + 1

freq.mat[k,1] <- j

freq.mat[k,2] <- sum(ifelse(mat==j,1,0))

freq.mat[k,3] <- sum(ifelse(mat==j,1,0))/tot

}

freq.mat

}

#-----------------------------------------------------------------

#Test Results

reps <- 10000

a <- .15

seed <- 0

f <- cbind(0, 0.3455439, 0.0357068, 0.0028667, 0.0009185) #CALCULATE IN MATHEMATICA© [21] FOR a

l <- cbind(0,0,0,0,0) #CALCULATE IN MATHEMATICA FOR a

ALIAS <- ALIAS.Method(reps, f, l, seed)

LS <- LS.Method(reps, a, seed)

LB <- LB.Method(reps, a, seed)

LBM <- LBM.Method(reps, a, seed)

LK <- LK.Method(reps, a, seed)

{-(a^1)/(1\*log(1-a))}\*10000

{-(a^2)/(2\*log(1-a))}\*10000

{-(a^3)/(3\*log(1-a))}\*10000

{-(a^4)/(4\*log(1-a))}\*10000

{-(a^5)/(5\*log(1-a))}\*10000

hist(ALIAS)

hist(LS)

hist(LB)

hist(LBM)

hist(LK)

freq(ALIAS)

summary(ALIAS)

sqrt(var(ALIAS))

freq(LS)

summary(LS)

sqrt(var(LS))

freq(LB)

summary(LB)

sqrt(var(LB))

freq(LBM)

summary(LBM)

sqrt(var(LBM))

freq(LK)

summary(LK)

sqrt(var(LK))

write.table(ALIAS, file = 'C:\\Perfomance Evaluation\\QAOM\\Thesis - Finitization\\Simulated Distributions\\AliasAlpha15.txt', row.names=FALSE, col.names=FALSE)

write.table(LS, file = 'C:\\Perfomance Evaluation\\QAOM\\Thesis - Finitization\\Simulated Distributions\\LS\_Alpha15.txt', row.names=FALSE, col.names=FALSE)

write.table(LB, file = 'C:\\Perfomance Evaluation\\QAOM\\Thesis - Finitization\\Simulated Distributions\\LB\_Alpha15.txt', row.names=FALSE, col.names=FALSE)

write.table(LBM, file = 'C:\\Perfomance Evaluation\\QAOM\\Thesis - Finitization\\Simulated Distributions\\LBM\_Alpha15.txt', row.names=FALSE, col.names=FALSE)

write.table(LK, file = 'C:\\Perfomance Evaluation\\QAOM\\Thesis - Finitization\\Simulated Distributions\\LK\_Alpha15.txt', row.names=FALSE, col.names=FALSE)

#-----------------------------------------------------------------

#Begin Horse Race

races <- 2

reps <- 1500000

a <- .15

seed <- 0

f <- cbind(0, 0.345543915, 0.035706843, 0.002866728, 0.000918505) #CALCULATE IN MATHEMATICA© [21] FOR a

l <- cbind(0,0,0,0,0) #CALCULATE IN MATHEMATICA© [21] FOR a

ALIAS.Time <- matrix(c(rep(0,races\*3)),nrow=races,ncol=3,dimnames=list(c(1:races),c(‘user’,’system’,’elapsed’)))

LS.Time <- matrix(c(rep(0,races\*3)),nrow=races,ncol=3,dimnames=list(c(1:races),c(‘user’,’system’,’elapsed’)))

LB.Time <- matrix(c(rep(0,races\*3)),nrow=races,ncol=3,dimnames=list(c(1:races),c(‘user’,’system’,’elapsed’)))

LBM.Time <- matrix(c(rep(0,races\*3)),nrow=races,ncol=3,dimnames=list(c(1:races),c(‘user’,’system’,’elapsed’)))

LK.Time <- matrix(c(rep(0,races\*3)),nrow=races,ncol=3,dimnames=list(c(1:races),c(‘user’,’system’,’elapsed’)))

for (z in 1:races)

{

#LS.Time[z,] <- matrix(system.time(LS.Method(reps, a, seed),gcFirst=TRUE)[1:3])

LB.Time[z,] <- matrix(system.time(LB.Method(reps, a, seed),gcFirst=TRUE)[1:3])

LBM.Time[z,] <- matrix(system.time(LBM.Method(reps, a, seed),gcFirst=TRUE)[1:3])

LK.Time[z,] <- matrix(system.time(LK.Method(reps, a, seed),gcFirst=TRUE)[1:3])

ALIAS.Time[z,] <- matrix(system.time(ALIAS.Method(reps, f, l, seed),gcFirst=TRUE)[1:3])

}

ALIAS.Time[1,’user’]

summary(ALIAS.Time[2:races,’user’])

#summary(LS.Time[2:races,’user’])

summary(LB.Time[2:races,’user’])

summary(LBM.Time[2:races,’user’])

summary(LK.Time[2:races,’user’])

summary(ALIAS.Time[2:races,’system’])

#summary(LS.Time[2:races,’system’])

summary(LB.Time[2:races,’system’])

summary(LBM.Time[2:races,’system’])

summary(LK.Time[2:races,’system’])

summary(ALIAS.Time[2:races,’elapsed’])

#summary(LS.Time[2:races,’elapsed’])

summary(LB.Time[2:races,’elapsed’])

summary(LBM.Time[2:races,’elapsed’])

summary(LK.Time[2:races,’elapsed’])**Appendix B: Mathematica© [21] Code for Generating the *NTSF* Probabilities for the Logarithmic Distribution**

**n = 4**

4

g[x\_]=Normal[Series[pxLog[1-p-x]/(p+x)Log[1-p]],{x,0,n}]]

1+x(-1/p+1/(-1+p)Log[1-p]) +

x4(1/p4 - 1/4(1-p)4Log[1-p] - 1/(-1+p)p2Log[1-p] - 1/2(1-p) 2p2Log[1-p] + 1/3(1-p)3pLog[1-p]) +

x3(-1/p3 -1/3(1-p)3Log[1-p] + 1/(-1+p)p2Log[1-p] + 1/2(1-p) 2pLog[1-p]) +

x2(1/p2 -1/2(1-p)2Log[1-p] - 1/(-1+p)pLog[1-p])

For[i=0,i≤n,Print[P[i]=(D[g[x],{x,i}]/.x→-p)pi/i!)//Simplify];i++]

p((48-162p+188p2-77p3)+60(-1+p)4Log[1-p])/12(-1+p)4Log[1-p]

p((-60+207p-248p2+107p3)-60(-1+p)4Log[1-p])/6(-1+p)4Log[1-p]

p((20-70p+86p2-39p3)+20(-1+p)4Log[1-p])/2(-1+p)4Log[1-p]

p((-30+105p-130p2+61p3)-30(-1+p)4Log[1-p])/6(-1+p)4Log[1-p]

p((12-42p+52p2-25p3)+12(-1+p)4Log[1-p])/12(-1+p)4Log[1-p]



Plot[P[n-1],{p,0,1/n}]



-Graphics-

FindRoot[P[n-1]== 0,{p,1/n}]

{p→0.239702}

p=.15

0.15

For[i=0, i ≤ n, Print[N[P[i],30]];i++]

0.922993

0.0691088

0.00714137

0.000573346

0.000183701