StatR 101: Fall 2012

Homework 7

Rod Doe

Saturday, November 10, 2012

# Trick or Treat

## Probability Mass Functions of Candy Distribution

par(mfrow=c(1,3))

x.a = 1:6

f.a = dunif(x.a, 0, 6)

plot(x.a, f.a, xlab="Abel", ylab="pmf", pch=19)

lines(x.a, f.a)

x.b = 0:6

f.b = dbinom(x.b, 6, 0.5)

plot(x.b, f.b, main="Candy Distribution", xlab="Bernoulli", ylab="pmf", type="h")

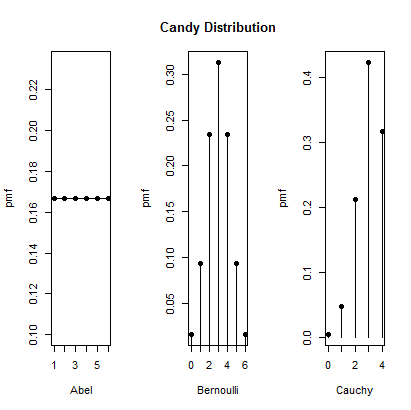
points(x.b, f.b, pch=19)

x.c <- 0:4

f.c <- dbinom(x.c, 4, 0.75)

plot(x.c, f.c, xlab="Cauchy", ylab="pmf",type="h")

points(x.c, f.c, pch=19)



## Name Distributions

Mr. Abel distributes candy using a uniform distribution ranging from 1 to 6. The set of results includes 1 to six pieces of candy.

Mrs. Bernoulli distributes candy according to a binomial distribution with n = 6 and probability of success = 0.5. The result set includes 0 to 6 pieces of candy.

Monsieur Cauchy distributes candy according to a binomial distribution with n = 4 and probability of success = 0.75. The result set include 0 to 4 pieces of candy.

## Expectations and Variances

|  |  |  |
| --- | --- | --- |
| **Name** | **Expectation** | **Variance** |
| Abel | 3.5 | 3 |
| Bernoulli | 3.0 | 1.5 |
| Cauchy | 3.0 | 0.75 |

Kids can expect the most candy from Mr. Abel. They can expect the least variance in candy count from Monsieur Cauchy.

x.a = 1:6

f.a = dunif(x.a, 0, 6)

(var.a = ((6-0)^2) / 12)

[1] 3

e.a = x.a \* f.a

(sum(e.a))

[1] 3.5

x.b = 0:6

f.b = dbinom(x.b, 6, 0.5)

(var.b=6\*0.5\*(1-0.5))

[1] 1.5

e.b = x.b \* f.b

(sum(e.b))

[1] 3

x.c <- 0:4

f.c <- dbinom(x.c, 4, 0.75)

(var.c=4\*0.75\*(1-0.75))

[1] 0.75

e.c = x.c \* f.c

(sum(e.c))

[1] 3

## Simulated Halloween for 10,000 kids

# Simulation of candy hauls.

x.a = 1:6

x.b = 0:1

x.c = 1:52

candy.a = 0

candy.b = 0

candy.c = 0

candy = numeric(10000)

for (i in 1:length(candy)) {

candy.a = 0

candy.b = 0

candy.c = 0

# Min value can be 1 (Abel only).

# Max value can be 6 (Abel) + 6 (Bernoulli) + 4 (Cauchy) = 16.

# Abel simulation is simply to choose a random number in 1:6.

# Failure is not possible.

candy.a = sample(x.a, 1, replace=TRUE)

# Bernoulli simulation is to choose a value between 0:1 6 times.

# Result is count of odd numbers (heads = 1, tails = 0).

# Successful if lands head.

candy.b = 0

for (b in 1:6) { candy.b = candy.b + sample(x.b, 1, replace=TRUE) }

# Cauchy simulation is to sample 1 value in 1:52 with replacement 4 times.

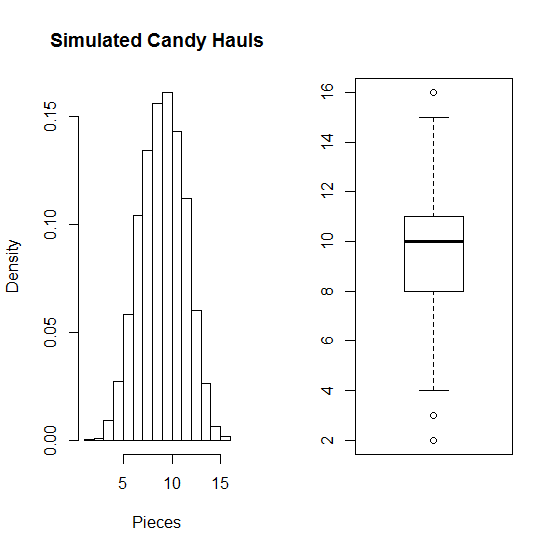
# Hearts are values 1:13.

# Successful if not a heart is drawn.

for (c in 1:4) { candy.c = candy.c + (sample(x.c, 1, replace=TRUE) > 13) }

candy[i] = candy.a + candy.b + candy.c

}



### Do the simulated numbers meet expectations?

mean(candy)

[1] 9.528

The expectations were (3.5, 3.0, 3.0), with a sum of 9.5. So, the simulated mean is near expectations.

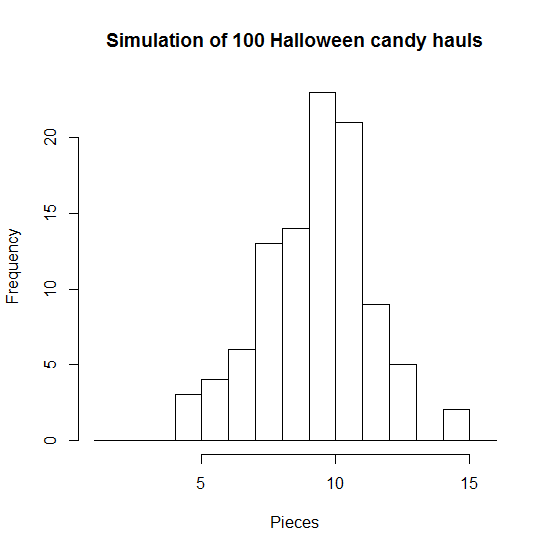
var(candy)

[1] 5.179134

From lecture, Var(A + B + C) = Var(A) + Var(B) + Var(C).

The variances were (3, 1.5, 0.75) with a sum of 5.25. So, the simulated variance looks realistic.

## Simulation of 100 Halloween hauls



This simulation yielded no hauls of less than 5 pieces, and 12 hauls of more than 12 pieces. As such, this simulated distribution is not symmetric.

> (length(which(candy < 5)))

[1] 0

> (length(which(candy > 12)))

[1] 7

# Earthquake Analysis

## Load and convert to minute differences

setwd("C:/Users/Rod/SkyDrive/R/101/Week07")

eq = read.csv("Earthquakes.csv", header=T)

str(eq)

Date <- as.POSIXlt(eq$Date)

Minute <- as.numeric(Date - min(Date))/60

W = diff(Minute)

W = diff(Minute) \* -1

Note: Minute is a vector of minutes between consecutive earthquakes.

## Mean, standard deviation, and variance of W

> mean(W)

[1] 39.85237

> sd(W)

[1] 43.78816

> var(W)

[1] 1917.403

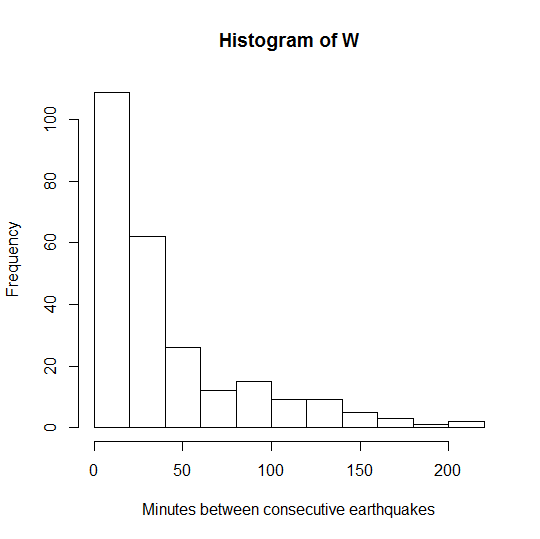
### Distribution

For exponential distribution:

* Rate = 1/lambda
* Mean = Rate
* Variance = Rate^2 (close but not perfect)

Var(W) is approximately mean(W) ^2, so this might be a good fit for an exponential distribution. The rate of the exponential distribution would be around 40.

## Histogram of W

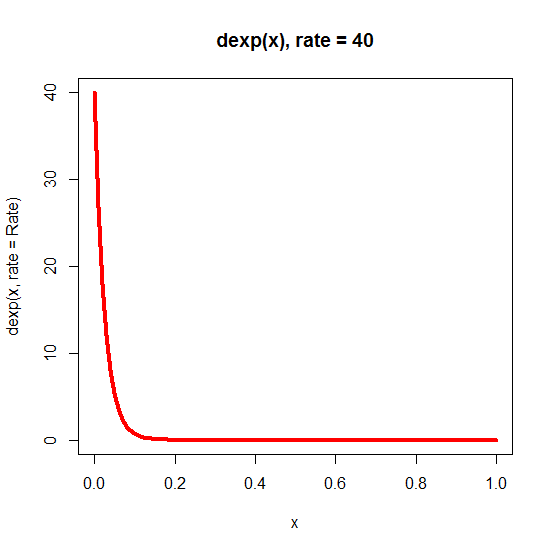


x= W

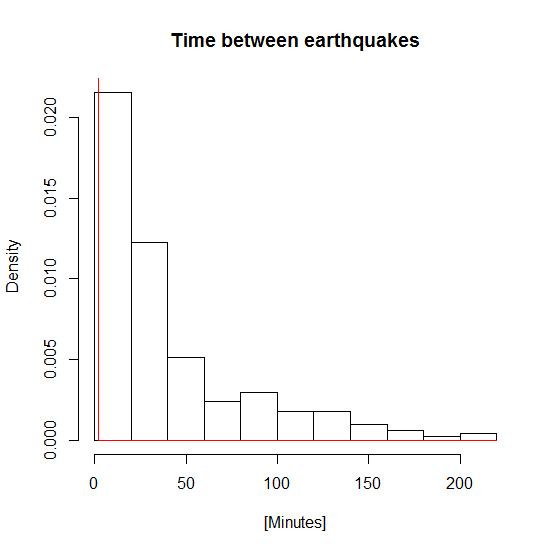
hist(x, prob=TRUE, main="Time between earthquakes", xlab="[Minutes]")

Rate=40

curve(dexp(x, rate=Rate), add=TRUE, col=rgb(1,0,0), lwd=4)



The red exponential distribution curve appears to correctly model the distribution of time between earthquakes. However, it appears to be distorted by scaling it to fit the histogram (see below).



The vector Minute is the minutes between consecutive quakes, with the final point being 0. The assignment requests calculation of quakes per hour. This requires removal of that pesky 0 to do the inversion.

which(Minute > 0)

QuakesPerMinute = 1/which(Minute > 0)

QuakesPerHour = 60 \* QuakesPerMinute

N.hour = QuakesPerHour

mean(N.hour)

[1] 1.449624

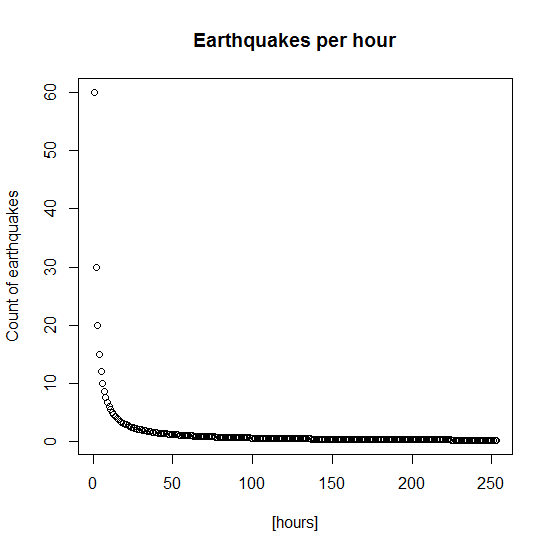
var(N.hour)

[1] 21.33296

sd(N.hour)

[1] 4.618761

plot(N.hour, main="Earthquakes per hour", xlab="[hours]", ylab="Count of earthquakes")



# Gamma function

## Gamma2PDF

Gamma2PDF = function(x, lambda) {

lambdasq = lambda \* lambda

d = numeric(length(x))

for (i in 1:length(x)) { d[i] = (x[i]/lambdasq)\*exp(-x[i]/lambda) }

return(d) }

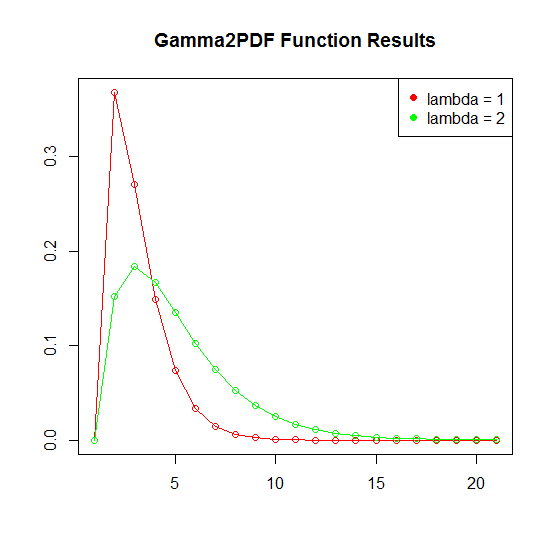
### Integration of Gamma2PDF should produce 1 if it is a valid PDF

> (integrate(Gamma2PDF, lower=0, upper=Inf, lambda=2))

1 with absolute error < 3.9e-06

It took some research to discover how to call the integrate function on the Gamma2PDF function, which takes two arguments. I Open Sourced my discovery on the class discussion board. Hopefully, somebody else can benefit from the research.

## Illustration of Gamma PDF



g1 = Gamma2PDF(0:20, 1)

g2 = Gamma2PDF(0:20, 2)

cols=c("red", "green")

plot(g1, col=cols[1], main="Gamma2PDF Function Results", xlab="", ylab="")

lines(g1, col=cols[1])

points(g2, col=cols[2])

lines(g2, col=cols[2])

legend("topright", legend=c("lambda = 1", "lambda = 2"), col=cols, pch=19)

I think I need to punt on this one. Out of time…