Solving questions with R

Gildas Taliah

0.1 Problem 1:

Consider an urn containing two red balls and eight green balls. We randomly draw ten balls one at a time, while putting it back each time before drawing again. What is the probability of drawing exactly two red balls?

```
# Define Parameters
N_red = 2 ; N_green = 8 ; N_trials = 10

# Technically this can be modeled with a binomial distribution,
# therefore we make use of 'dbinom' function

dbinom(x= 2, size= 10, prob= N_red/(N_red+N_green))
```

```
## [1] 0.3019899
```

What is the probability of drawing two or less red balls?

```
# Here, we make use of the cumulative function 'pbinom'

pbinom(q= 2, size= 10, prob= N_red/(N_red+N_green), lower.tail= T)
```

```
## [1] 0.6777995
```

Now consider an urn that contains 20 balls, including 10 red and 10 green balls. We randomly select 6 balls without replacement. What is a probability of getting two red balls?

```
# Define Parameters
N_r = 10 ;N_g = 10 ;N_t = 6

# This can be statistically modeled with a hypergeometric distribution
# therefore we make use of 'dhper' function

dhyper(x= 2, m= N_r, n= N_g, k= N_t)
```

```
## [1] 0.243808
```

In our last example, we consider an urn with two red, three green, and five blue balls. We randomly draw three balls with replacement. What is the probability of drawing one ball of each colour?

```
# This can be statistically modeled by a multinomial distribution
# therefore we make use of 'dmultinom' function

dmultinom(x= c(1,1,1),  prob=c(.2, .3, .5))
```

```
## [1] 0.18
```

0.2 Problem 2:

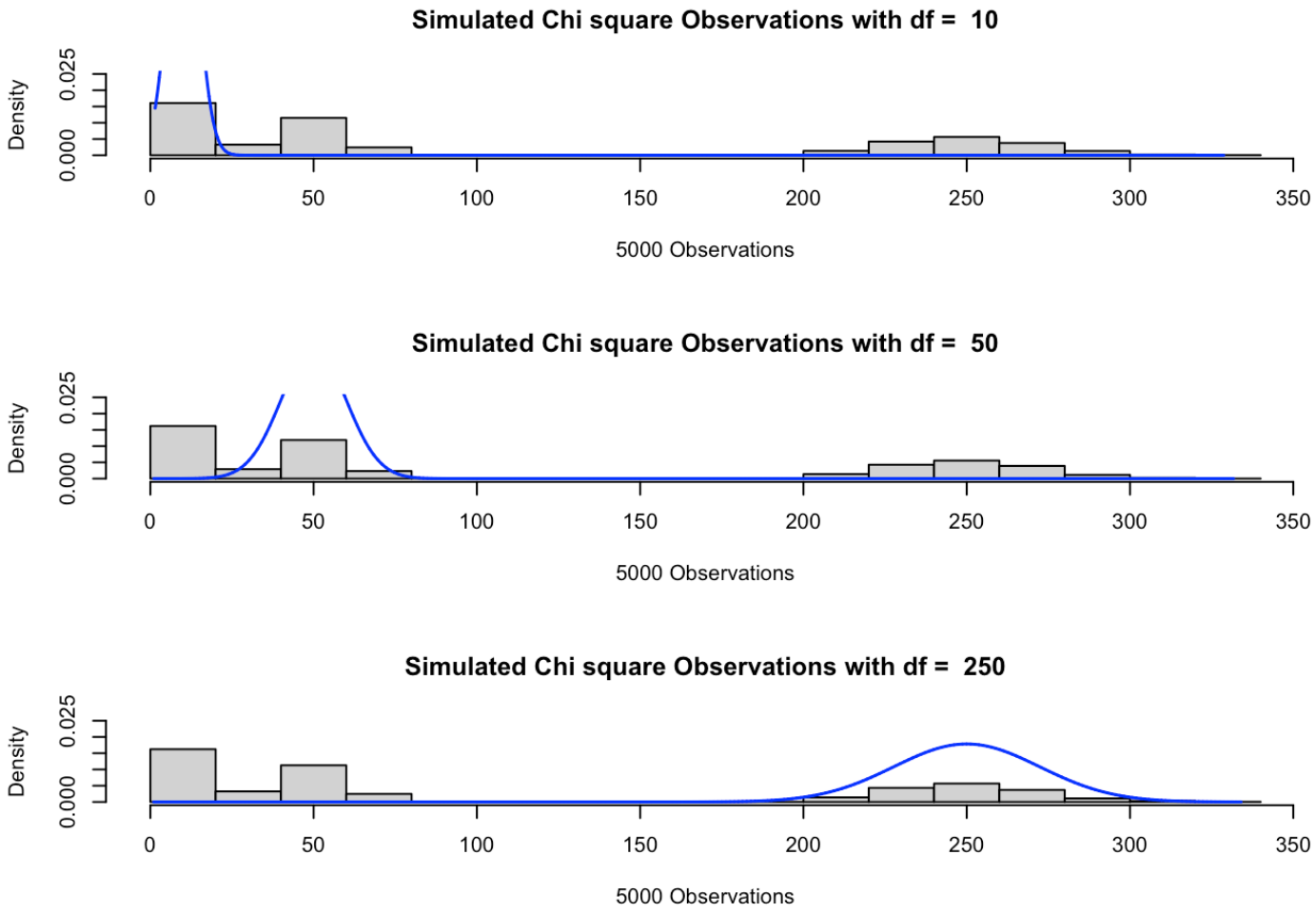
In this exercise we need to demonstrate that if a random variable X is χ^2-distributed with k degrees of freedom, then for $k \to \infty$ $X \xrightarrow{d} \mathcal{N}(k, 2k)$.

To this end, write a function that samples and returns N observations from the χ^2 distribution with k degrees of freedom. The function's input shall be k.

```
# Write a function that samples N observations from the chi-squared distribu
tion
# with k degrees of freedom.

chisquaredRand = function(K){
  # Function to Draw and return Sample
  sample = rchisq(N, df = k)
  return(sample)
}
```

Plot the resulting observations from the χ^2 distribution using histograms with Scott breaks for k = c(10, 50, 250), N = 5000 in a 3x1 plot window and add the respective limiting Normal density for comparison. Choose a meaningful title and label the axes.

```
## Define input parameters
N = 5000 #NMC
k = c(10, 50, 250)

## Generating Samples
set.seed(1200)
sample = matrix(0, nrow = N, ncol = length(k))

# Running a for loop
for (i in 1:length(k)){
  sample[,i] = chisquaredRand(k[i])
}

## Plotting the histograms
par(mfrow=c(3,1)) # Put histogram in 3 rows, 1 column
for(i in 1:length(k)){
  # Chi sqaure Observation with increasing degrees of freedom
  hist(sample[,i], probability = T, breaks = 'Scott', xlab= " 5000 Observati
ons ",
  ylab="Density", ylim = c(0,0.025)  ,main =paste('Simulated Chi square Obse
rvations with df = ', k[i]))
  # Plotting the Limiting 'Normal Density' for comparison
  x = seq(min(sample[,i]), max(sample[,i]), 0.01)
  lines(x, dnorm(x, mean= k[i], sd = sqrt(2*k[i])), col= 'blue', lwd= 1.5)
}
```




```
par(mfrow=c(1,1)) # Restore Default settings
```

0.3 Problem 3:

0.3.1 a)

Let the random variable X_i following a Chi Square distribution

$$f(x) = \frac{1}{2^{\frac{\nu}{2}}\Gamma\left(\frac{\nu}{2}\right)} x^{\frac{\nu}{2}-1} e^{-\frac{x}{2}} \mathcal{I}_{(0,\,\infty)}(x)$$

have stochastically independent realizations.

Question: Find the asymptotic distribution of $\overline{X}_n^2$.

Answer: We know that X_i is stochastically independent, and to find the asymptotic distribution, we can make use of the Delta Method. Since X_i follows a Chi Square distribution, and following the reasoning of theorems 4.2 and 4.3, therefore we have $\overline{X}_n$ would eventually result to $\overline{X}_n \overset{a}{\sim} N(\nu, \frac{2\nu}{n})$. (eqn1).

Moreover, the Delta method stipulates that, if X_i be a sequence s.t $\sqrt{n}(X_n - \mu) \overset{d}{\sim} N(0, \Sigma)$. Then $\sqrt{n}(g(X_n) - g(\mu)) \overset{d}{\sim} N(0, G\Sigma G)$

applying this to eqn1

i.e We subtract the mean ν and multiply by $\sqrt{n}$, then we have we have $\sqrt{n}(\overline{X}_n - \nu) \overset{d}{\sim} N(0, 2\nu)$ Now, since we are concerned with the asymptotic distribution of $\overline{X}_n^2 = g(\overline{X})$.
…… eqn2 We proceed to take the first derivative of eqn2, evaluated at the mean and result to $2\nu \; i.e$ $ G = 2$,and eqn2 evaluated at the mean results to $\nu^2 \; i.e \; g(\mu) = \nu^2$

Taking all that into consideration and simplifying further, we result to $\sqrt{n}(\overline{X}_n^2 - \nu^2) \overset{d}{\sim} N(0, 8\nu^3)$ …. eqn3 Now, adding the mean ν^2 and dividing by $\sqrt{n}$ to eqn3, we therefore result to the asymptotic distribution of $\overline{X}_n^2$, which is : $\overline{X}_n^2 \overset{d}{\sim} N(\nu^2, \frac{8\nu^3}{n})$

Stats technique used: Delta Method.

0.3.2 b)

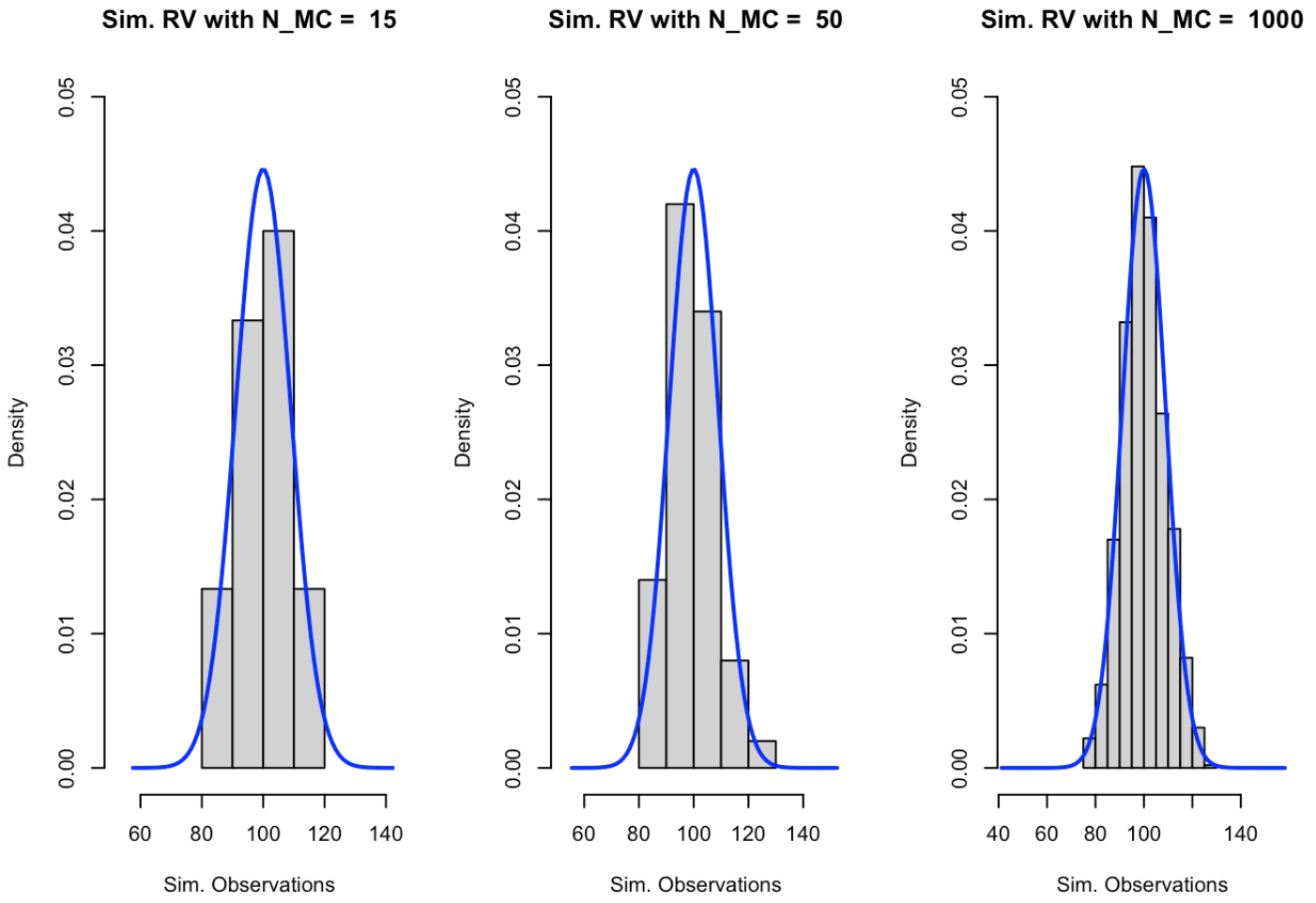
Write a function that simulates and returns N_{MC} squared sample means with sample size n for $X_i \sim \chi^2_{(\nu)}$ where $\nu = 10$. The function's inputs shall be N_MC and n. *Hint:* In order to avoid a for loop, the function **rowMeans()** might be helpful.

```
means_square_Chisq = function(N, n){
  sample = replicate(n = n, rchisq(n = N, df = 10))
  means_square_Chisq= (rowMeans(sample))^2
  return(means_square_Chisq)

}
```

0.3.3 c)

Plot the resulting squared means using histograms with Scott breaks for N_MC = c(15, 50, 1000), n = 100 in a 1x3 plot window and add the respective limiting Normal density for comparison.

```
# Define input parameters
N_MC= c(15, 50, 1000)
n = 100
nu = 10
## Set seed for distinct forthcoming random Samples
set.seed(125)


## Plotting  histograms
par(mfrow=c(1,3)) # Put histogram in 1 row, 3 columns
for(i in 1:length(N_MC)){
  # Mean Chi sqaured Observation with increasing Number of samples
  hist(means_square_Chisq(N_MC[i], n), probability = T, breaks = 'Scott',
       xlab= "Sim. Observations", ylab= "Density", ylim = c(0,0.05),
       xlim = c(min(means_square_Chisq(N_MC[i],n))-30,max(means_square_Chisq
(N_MC[i], n))+30), main =paste('Sim. RV with N_MC = ', N_MC[i]))
  # Plotting the Limiting 'Normal Density' for comparison
  x = seq(min(means_square_Chisq(N_MC[i], n))-30, max(means_square_Chisq(N_M
C[i], n)+30))
  lines(x, dnorm(x, mean= nu^2 , sd = sqrt((8*nu^3)/n)), col= 'blue', lwd=2)
}
```




```
par(mfrow=c(1,1)) # Restore Default settings
```

0.4 Problem 4:

0.4.1 a)

Write a function that computes and returns (as a list!) the first four central moments of any given sample. We are NOT allowed to use any built-in functions for computing the moments, except **mean()**.

```
Moments = function(x){
  # Maean
  mean = mean(x)
  #Variance
  sum_var=0
  for (i in 1:length(x)){
    sum_var = sum_var +  (x[i]-mean)^2
  }
  var = sum_var/length(x)
  #Skewness
  sum_skw=0
  for (i in 1:length(x)){
    sum_skw = sum_skw +  (x[i]-mean)^3
  }
  skw = sum_skw/length(x)
  # Kurtosis
  sum_kts=0
  for (i in 1:length(x)){
    sum_kts = sum_kts +  (x[i]-mean)^4
  }
  kts = sum_kts/length(x)
  # Returns a list of Moments
  return(list(Mean=mean, Variance= var , Skewness =skw, Kurtosis=kts))
}
```

0.4.2 b)

Prove that if you simulate a random sample a large number of times and compute the average of the moments of all the simulated samples, the resulting moment values will converge to the true moment values.

To do this, simulate N_MC=1000 samples of size n=100 from a Standard Normal distribution, compute the first four central moments for each sample using the previously generated function **Moments()** (from part a), compute the average of each moment across all simulated samples. Do this without any loop.

```
# Define input parameters
N_MC = 1000
n = 100

# Generate samples and compute moments
set.seed(150)
sample = replicate(n=n, rnorm(n=N_MC, mean = 0, sd = 1))
dim(sample)
```

```
## [1] 1000  100
```

```
# Compute the average of the moments across all samples
# making use of the function 'apply', to compute the moments of each 'nth' d
ata set with its's 1000 observations
Moms = apply(X= sample, MARGIN = 2, FUN = Moments)

## Unlisting the nested list, and saving in a data frame
unlisted = data.frame(do.call(rbind.data.frame, Moms))


## computing the average of each set of Moments
means = apply(X= unlisted, MARGIN =2, FUN = mean)

#Printing the respective average computed moments and rounding to nearest va
lues
Av_moms = as.data.frame(cbind2(means, round(means)))
names(Av_moms)[1]= 'Computed Avrg Moments' ; names(Av_moms)[2]= 'Approximate
d Avrg Moments'
print(Av_moms)
```

```
##          Computed Avrg Moments Approximated Avrg Moments
## Mean               0.002270330                         0
## Variance           0.996531511                         1
## Skewness           0.005611556                         0
## Kurtosis           2.960996150                         3
```

What do you conclude?

Comments: As observed, the computed average moments with large data sample, converges to the true Moments of the underlying distribution. This is evident from the approximated values which results to the exact Moments values of a Standard Normal distribution $X \sim N(0, 1)$. i.e Mean = 0, variance = 1, Skewness = 0 and Kurtosis = 3.

0.5 Problem 5:

0.5.1 a)

Write a function that computes and returns (as a matrix, where each column corresponds to a certain moment!) partial mean, partial variance, partial skewness and partial kurtosis of a sample.

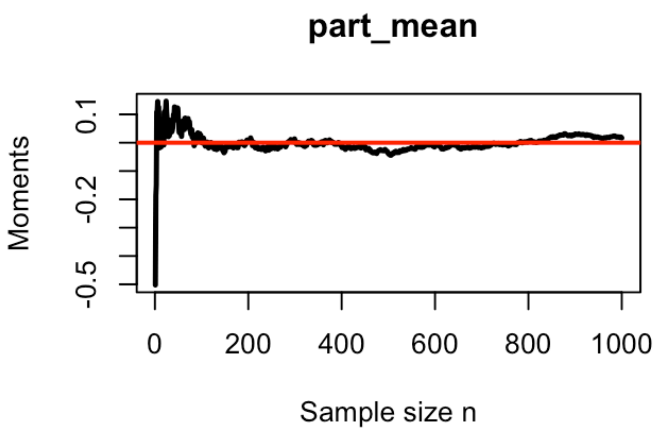
Recall: - the partial mean of a sample $X_1, \ldots, X_n$ is defined by $\bar{x}_j = \frac{1}{j} \sum_{i=1}^{j} x_i$ - the partial variance is defined by $\hat{\sigma}_j^2 = \frac{1}{j} \sum_{i=1}^{j} (x_i - \bar{x}_i)^2$ - the partial skewness is defined by $\hat{\mu}_{3,j} = \frac{1}{j} \sum_{i=1}^{j} (x_i - \bar{x}_i)^3$ - the partial kurtosis is defined by $\hat{\mu}_{4,j} = \frac{1}{j} \sum_{i=1}^{j} (x_i - \bar{x}_i)^4$ Do not use any loops!

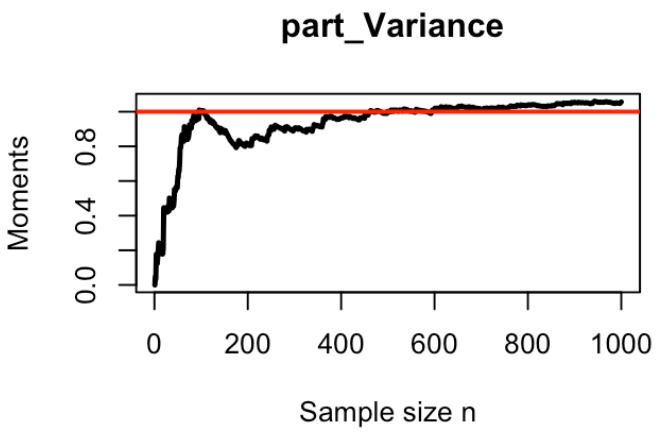
```
PartMoments = function(x){
  means= cumsum(x)/(1:length(x))
  variance = cumsum((x-cumsum(x)/(1:length(x)))^2)/(1:length(x))
  means_03= cumsum((x-cumsum(x)/(1:length(x)))^3)/(1:length(x))
  means_04= cumsum((x-cumsum(x)/(1:length(x)))^4)/(1:length(x))
  return (cbind(means, variance, means_03, means_04))
}
```

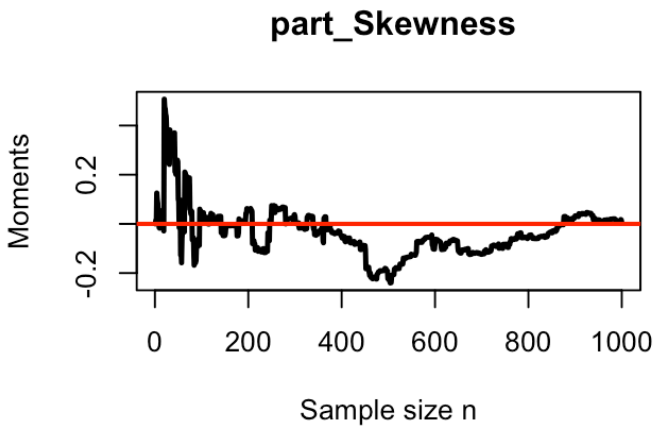
0.5.2 b)

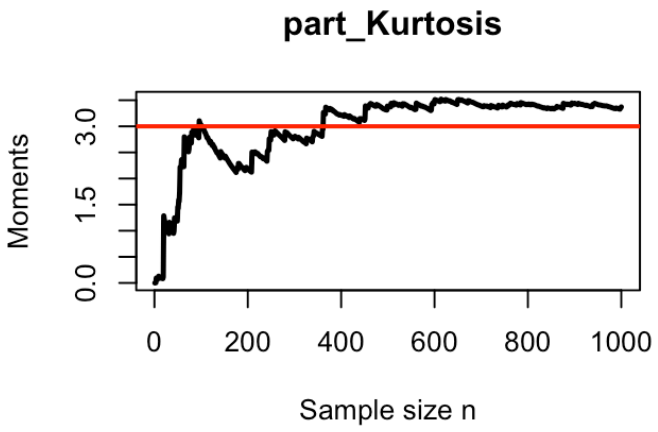
Generate a sample of size n=1000 from a Standard Normal distribution. Using the function generated in part (a) compute and plot (in a 2x2 plot window) the partial moments of that sample. Add a red line to each plot indicating the true moment value using the function **abline()**. Choose a meaningful title and label the axes.

```
#Draw sample
n = 1000
set.seed(100)
sam= rnorm(n, mean=0, sd= 1)
names = c('part_mean', 'part_Variance', 'part_Skewness','part_Kurtosis')
Pt_Mmts= PartMoments(sam)
mom = c(0,1,0,3)

par(mfrow=c(2,2)) # Put histogram in 2 rows, 2 column
for(i in 1:4){
  # Plotting respective moments
  plot(1:n , Pt_Mmts[,i] , type= 'l' ,lwd=2.5, xlab = 'Sample size n' , ylab
= 'Moments' , main =paste('',names[i] ))
  # A plot for Comparison
  abline(h= mom[i], col= 'red', lwd= 2)
}
```










```
par(mfrow=c(1,1)) # Restore Defaut setting
```

0.6 Problem 6 :

Show that the standardized Poisson distribution can be approximated by the Standard Normal distribution for large λ.

0.6.1 a)

Write a function that generates random samples from the Poisson distribution and takes λ as an argument.

```
PoisSample = function(lambda){
  # Function to Draw and return poison random samples
  draws = rpois(n, lambda = lambda)
  return(draws)
}
```

0.6.2 b)

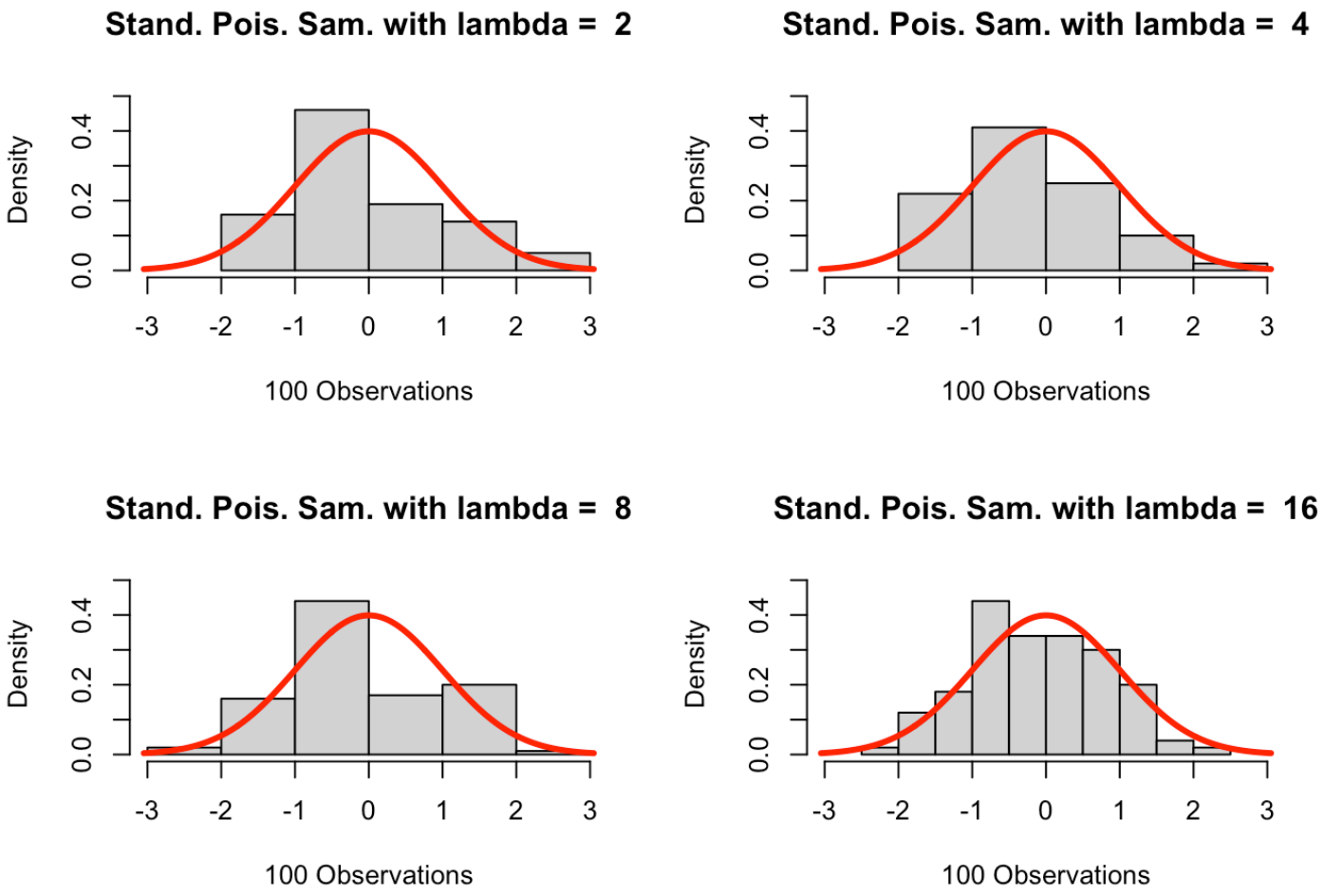
Write a function that standardizes the sample.

```
StandardSample = function(x, lambda){
  # Function to Standardize (and return) poisson generated samples
  z = (x-lambda)/sqrt(lambda)
  return(z)
}
```

0.6.3 c)

Use the above generated functions to plot four histograms (in a 2x2 plot window) of the standardized Poisson sample of size n=100 for four different values of λ, namely $\lambda = c(2, 4, 8, 16)$. Set the x-axis limits to be c(-3,3). Use the “Scott” method for breaks parameter. On top of each histogram add a Standard Normal distribution in red color. Choose a meaningful title and label the axes. Comment on the result.

```
# Define input parameters
n= 100
lambda = c(2,4,8,16)

# Set seed and obtain Zero filled matrices
set.seed(125)
mat_samp= matrix(0, nrow = n, ncol=length(lambda))
mat_sta_samp= matrix(0, nrow = n, ncol=length(lambda))

# Constructing for loop to draw poisson samples
for (i in 1:length(lambda)){
  mat_samp[,i]= PoisSample(lambda[i])

}
# Building a for loop to Standardize poison samples
for (i in 1:length(lambda)){
  mat_sta_samp[,i] = StandardSample(mat_samp[,i], lambda[i])
}
#print(mat_sta_samp)

# Plot
par(mfrow=c(2,2)) # Putt histogram in 2 rows, 2 columns
for(i in 1:4){
  # Empirical
  hist(mat_sta_samp[,i], probability = T, breaks = 'Scott', xlab= " 100 Obse
rvations ", ylab=" Density", main =paste('Stand. Pois. Sam. with lambda = ',
lambda[i]) , xlim = c(-3,3), ylim = c(0,0.5))
  # Adding respective Limiting 'Normal Density' for comparison
  x = seq(-3.05, 3.05, 0.01)
  lines(x, dnorm(x, mean= 0, sd = 1), col= 'red', lwd = 3)
}
```




```
par(mfrow=c(1,1)) # Restore default setting
```

What do you conclude?

Comments: The output produces the desired results for standardized values, as observed, an increasing lambda leads to a more desired result, which serves as evidence to the claim, a poisson distribution can be approximated by the normal distribution for large lambda $\lambda \to \infty$.

0.7 Problem 7:

0.7.1 a)

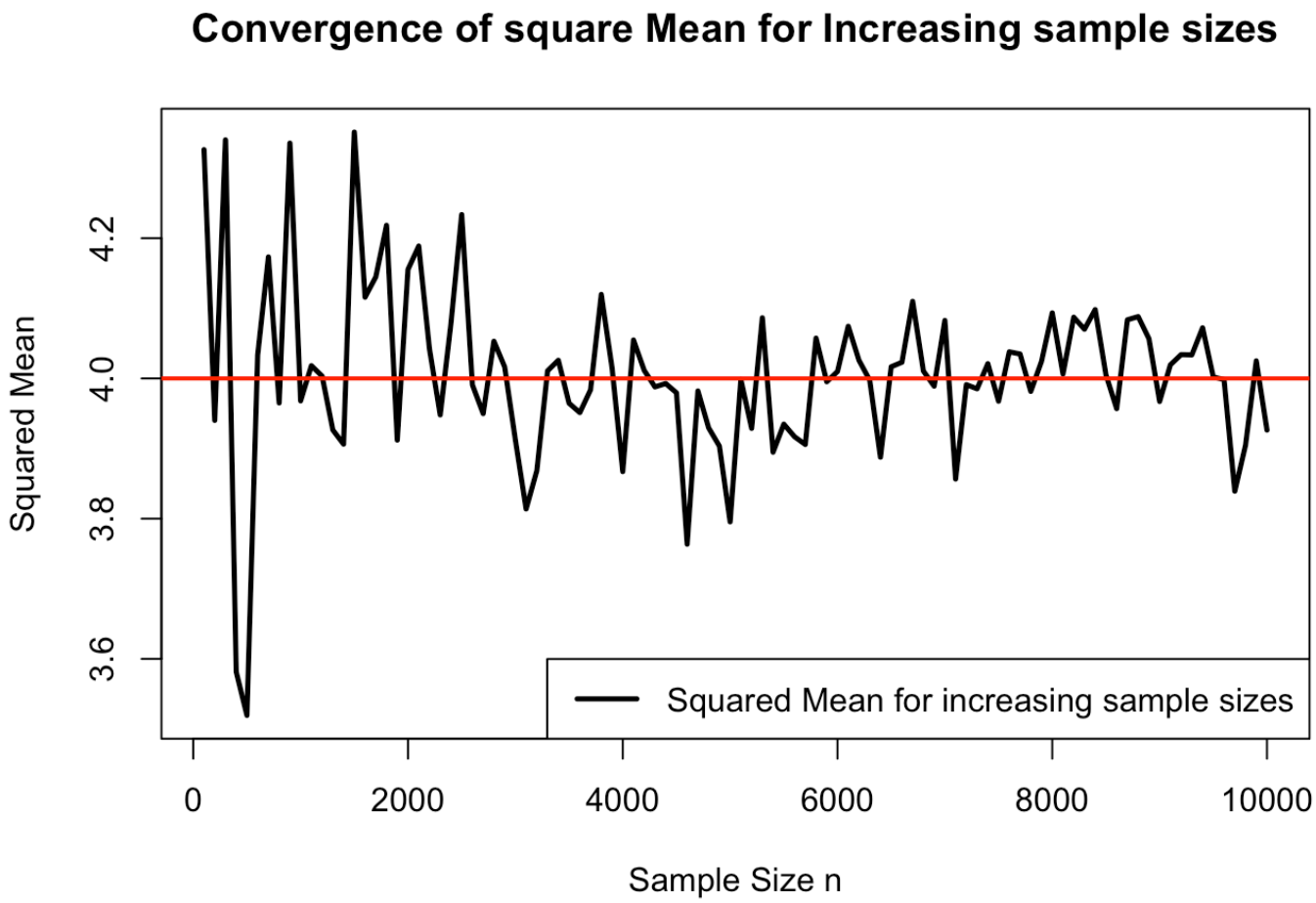
Write a function that takes a sample size n as an argument, samples from a Poisson distribution and computes and returns the squared mean $\bar{X}_n^2 = \left(\frac{\sum_{i=1}^{n} X_i}{n} \right)^2$.

```
SqMean = function(n){
  sample = rpois(n,lambda = lambda )
  sqmean = (mean(sample))^2
  return(sqmean)
}
```

0.7.2 b)

Generate a sequence of n values that goes from 100 to 10000 by 100. For each n from the sequence compute a squared mean using previously generated function **SqMean()** and store the value in a vector. Plot the resulting vector of squared means against n. Add a red line for the true value λ^2 to the plot using the function **abline()**. Choose a meaningful title and label the axes. Comment on the result.

```
# Define input parameters
lambda = 2

# Set seed, acquire sequence and (generate random numbers and) compute squar
e Means
set.seed(125)
seq = seq(from = 100, to = 10000, by = 100)
vec = mat.or.vec(length(seq), 1)

# Building a for loop to compute "Squared Mean' for increasing samples
for(i in 1:length(seq)){
  vec[i]=SqMean(seq[i])
}


# Plot squared mean against increasing sample sizes
plot( seq, vec, lwd=2.5, type= 'l', xlab = 'Sample Size n', ylab = "Squared
Mean", main = 'Convergence of square Mean for Increasing sample sizes')
abline(h= lambda^2, col= 'red', lwd= 2)
legend('bottomright', paste('Squared Mean for increasing sample sizes'), lwd
= 2.5, col= 'black')
```




```
print('The End!')
```

```
## [1] "The End!"
```

Comment: As sample size increases, the squared means are much more closer to the true underlying square mean. i.e ($n \to \infty$), the sample's squared mean converges to the true underlying sqaured mean λ^2.

This tasks were solved by me and there could exist some incorrectness.