9 Additional Graphs Homework

7-31-20

General Instructions

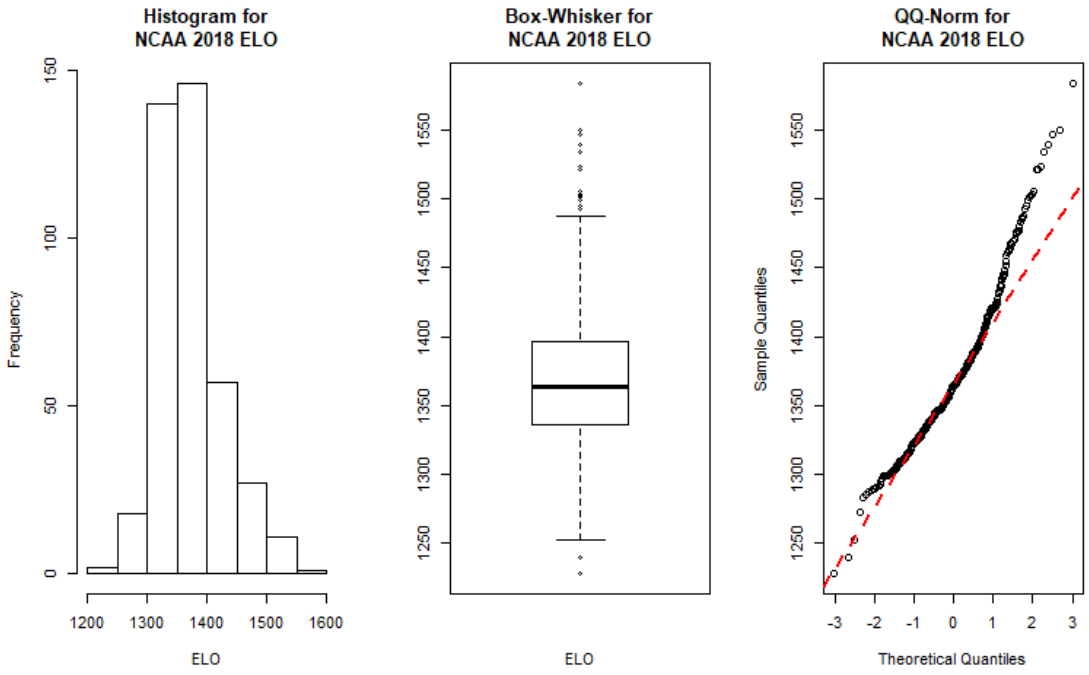
There are six exercises below. You are required to provide five solutions, with the same options for choosing languages as with the last exercise. You can provide solutions in two languages for one exercise only (for example, Ex. 1,2,3,5 in R and Ex. 1 in SAS is acceptable, Ex. 1,2,3 in SAS and Ex. 1,2 in R is not)

For this exercise, you may use whatever graphics library you desire.

Exercise 1.

Load the `ncaa2018.csv` data set and create histograms, QQ-norm and box-whisker plots for `ELO`. Add a title to each plot, identifying the data.

```
elo.dat <- read.csv('ncaa2018.csv')
elo <- elo.dat$ELO

par(mfrow=c(1,3))
hist(elo, main = paste0('Histogram for','\n', 'NCAA 2018 ELO'),xlab = 'ELO')
boxplot(elo, main = paste0('Box-Whisker for','\n','NCAA 2018 ELO'),xlab =
'ELO')
qqnorm(elo,main = paste0('QQ-Norm for','\n','NCAA 2018 ELO'))
qqline(elo, col = 2,lwd=2,lty=2)
```

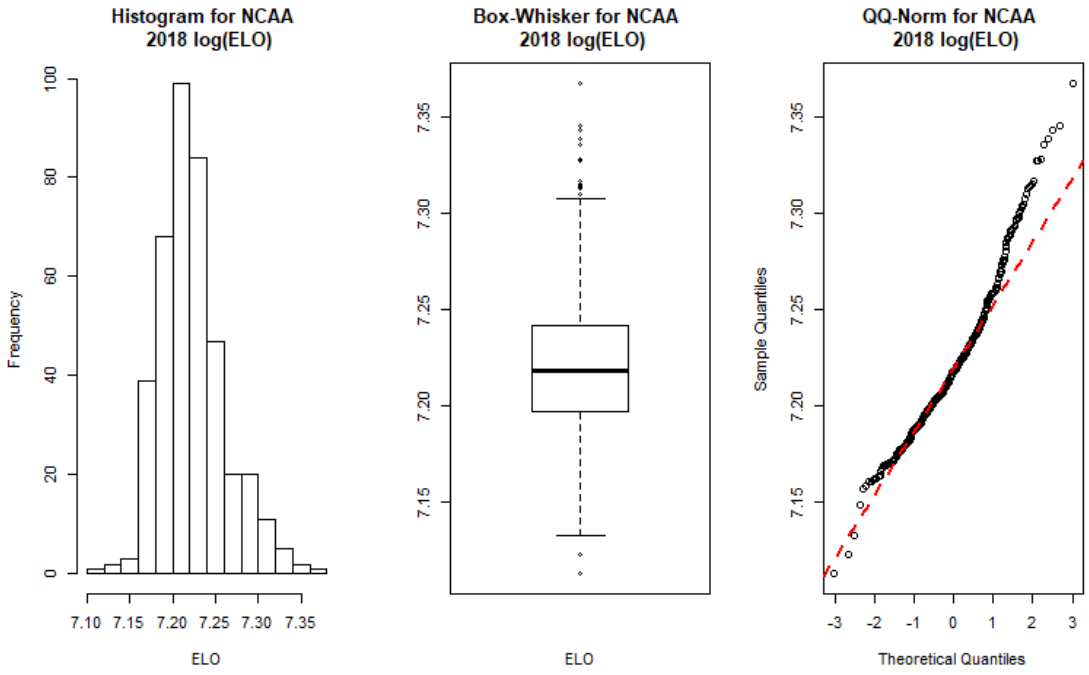


Part b

A common recommendation to address issues of non-normality is to transform data to correct for skewness. One common transformation is the log transform.

Transform `ELO` to `log(ELO)` and produce histograms, box-whisker and qqnorm plots of the transformed values. Are the transformed values more or less skewed than the original? You might calculate skewness and kurtosis values as in Homework 6, Exercise 2.

```
log.elo <- log(elo.dat$ELO)

par(mfrow=c(1,3))
hist(log.elo, main = 'Histogram for NCAA \n 2018 log(ELO)',xlab = 'ELO')
boxplot(log.elo, main = 'Box-Whisker for NCAA \n 2018 log(ELO)',xlab = 'ELO')
qqnorm(log.elo,main = 'QQ-Norm for NCAA \n 2018 log(ELO)')
qqline(log.elo, col = 2,lwd=2,lty=2)
```




```
skew_kurt <- function(x){
  temp.skew = 0
  temp.kurt = 0
  temp.sd = 0
  n=0
  for(i in 1:length(x)){
    if(!is.na(x[i])){
      mean_x <- mean(x,na.rm = TRUE)
      temp.sd <- temp.sd + (x[i] - mean_x)^2
      temp.skew <- temp.skew + (x[i] - mean_x)^3
      temp.kurt <- temp.kurt + (x[i] - mean_x)^4
      n <- n + 1
    }
  }
  sd_x <- sqrt(sum(temp.sd)/n)
  skew <- (sum(temp.skew)/n)/(sd_x^3)
  kurt <- (sum(temp.kurt)/n)/(sd_x^4)
  return(list(Skewness=skew,Kurtosis=kurt))
}

cat('Skewness and Kurtosis for ELO','\n')

## Skewness and Kurtosis for ELO

skew_kurt(elo)

## $Skewness
## [1] 0.8692405
##
```

```
## $Kurtosis
## [1] 4.032922

cat('Skewness and Kurtosis for log of ELO','\n')

## Skewness and Kurtosis for log of ELO

skew_kurt(log.elo)

## $Skewness
## [1] 0.7371095
##
## $Kurtosis
## [1] 3.787412

cat(' Skewness improvement is',
    round(as.numeric(skew_kurt(elo)[1])-as.numeric(skew_kurt(log.elo)[1]),4),
    '\n','Kurtosis improvement is',
    round(as.numeric(skew_kurt(elo)[2])-as.numeric(skew_kurt(log.elo)[2]),4))

##  Skewness improvement is 0.1321
##  Kurtosis improvement is 0.2455
```

Comment - The log of elo did improve skewness and kurtosis marginally. From 0.86 to 0.74 skewness and 4.03 to 3.79 kurtosis. The data is right skewed, however both elo and log(elo) are both under 1 so they are not severely skewed. The kurtosis shows the data is not overly or underly peaked, but may not be normally distributed.

Exercise 2.

Review Exercise 2, Homework 6, where you calculated skewness and kurtosis. The reference for this exercise,
https://www.itl.nist.gov/div898/handbook/eda/section3/eda35b.htm,

The following example shows histograms for 10,000 random numbers generated from a normal, a double exponential, a Cauchy, and a Weibull distribution.

We will reproduce the histograms for these samples, and add qqnorm and box-whisker plots.

Part a

Use the code below from lecture to draw 10000 samples from the normal distribution.

```
norm.sample <- rnorm(10000, mean=0, sd=1)
```

Look up the corresponding `r*` functions in R for the Cauchy distribution (use location=0, scale=1), and the Weibull distribution (use shape = 1.5). For the double exponential, use you can use the `*laplace` functions from the `rmutil` library, or you can use `rexp(10000) - rexp(10000)`

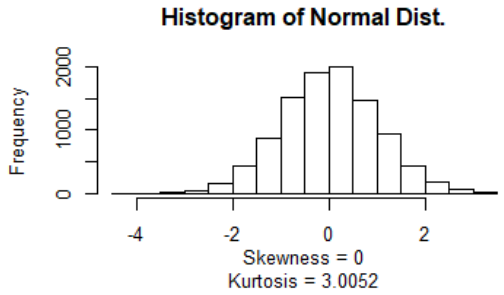
Draw 10000 samples from each of these distributions. Calculate skewness and kurtosis for each sample. You may use your own function, or use the `moments` library.

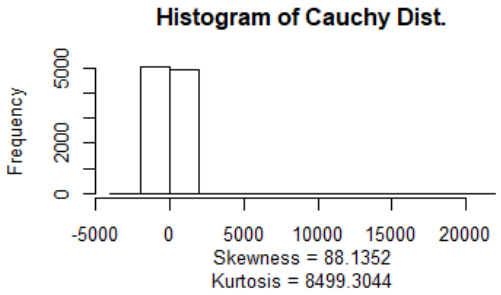
```
set.seed(111111)
sample.dcauchy <- rcauchy(1:10000)
cat('Skewness and Kurtosis for Cauchy Distribution','\n')

## Skewness and Kurtosis for Cauchy Distribution

skew_kurt(sample.dcauchy)

## $Skewness
## [1] 88.13518
##
## $Kurtosis
## [1] 8499.304

cat('\n')

sample.dweibull <- rweibull(1:10000,shape = 1.5)
cat('Skewness and Kurtosis for Weibull Distribution','\n')

## Skewness and Kurtosis for Weibull Distribution

skew_kurt(sample.dweibull)

## $Skewness
## [1] 1.083176
##
## $Kurtosis
## [1] 4.443246

cat('\n')

# install.packages('rmutil')
suppressWarnings(suppressMessages(library(rmutil)))

sample.dexp <- rlaplace(1:10000)
cat('Skewness and Kurtosis for Double Exponential','\n')

## Skewness and Kurtosis for Double Exponential

skew_kurt(sample.dexp)

## $Skewness
## [1] -0.1030751
##
## $Kurtosis
## [1] 5.8354
```

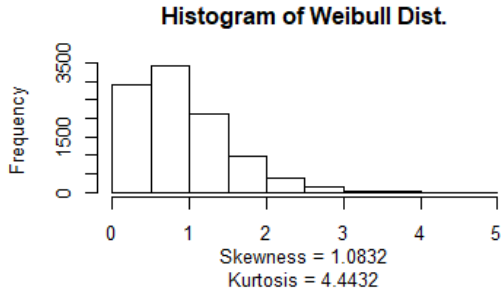
Part b

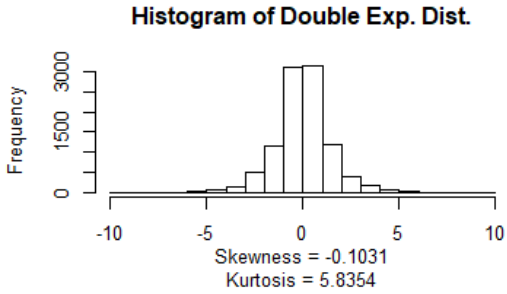
Plot the histograms for each distribution. Use `par(mfrow=c(2,2))` in your code chunk to combine the four histogram in a single plot. Add titles to the histograms indicating the distribution. Set the x-axis label to show the calculated skewness and kurtosis, i.e. `skewness = ####, kurtosis = ####`

```
par(mfrow=c(2,2))
sk_normdist <- skew_kurt(norm.sample)
hist(norm.sample,main = 'Histogram of Normal Dist.',xlab = paste0('Skewness =
',
     round(as.numeric(sk_normdist[1]),4),'\n','Kurtosis = ',
     round(as.numeric(sk_normdist[2]),4)))

sk_cauchy <- skew_kurt(sample.dcauchy)
hist(sample.dcauchy,main = 'Histogram of Cauchy Dist.',xlab =
paste0('Skewness = ',
     round(as.numeric(sk_cauchy[1]),4),'\n','Kurtosis = ',
     round(as.numeric(sk_cauchy[2]),4)))

sk_weibull <- skew_kurt(sample.dweibull)
hist(sample.dweibull,main = 'Histogram of Weibull Dist.',xlab =
paste0('Skewness = ',
     round(as.numeric(sk_weibull[1]),4),'\n','Kurtosis = ',
     round(as.numeric(sk_weibull[2]),4)))

sk_dexp <- skew_kurt(sample.dexp)
hist(sample.dexp,main = 'Histogram of Double Exp. Dist.',xlab =
paste0('Skewness = ',
     round(as.numeric(sk_dexp[1]),4),'\n','Kurtosis = ',
     round(as.numeric(sk_dexp[2]),4)))
```







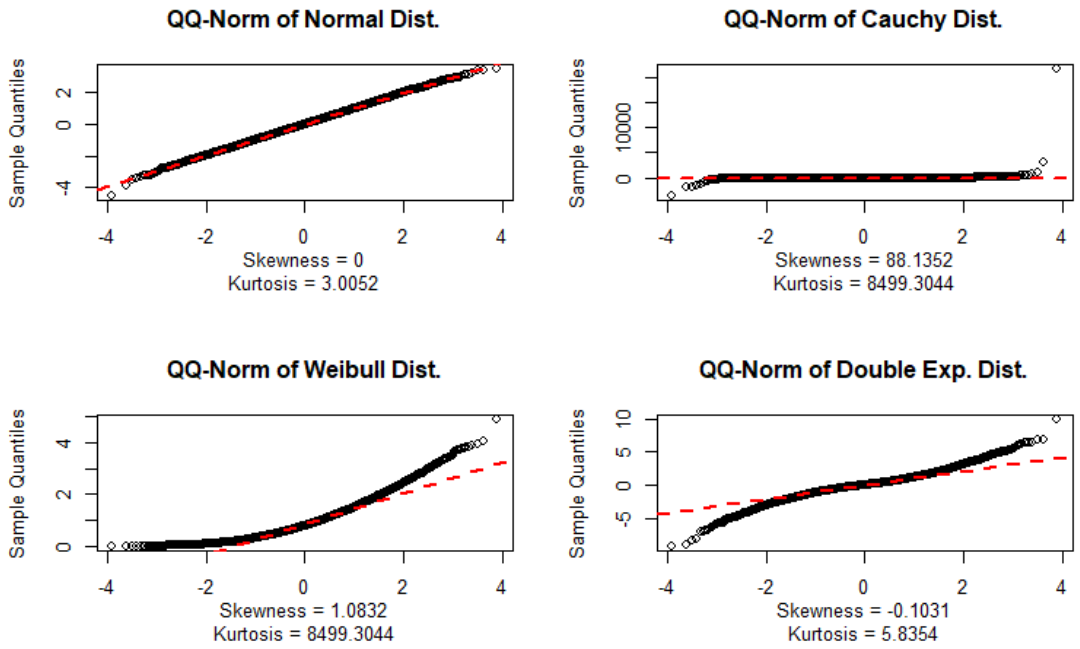


Part c

Repeat Part b, but with QQ-norm plots.

```
par(mfrow=c(2,2))

qqnorm(norm.sample,main = 'QQ-Norm of Normal Dist.',xlab = paste0('Skewness =
',
       round(as.numeric(sk_normdist[1]),4),'\n','Kurtosis = ',
       round(as.numeric(sk_normdist[2]),4)))
qqline(norm.sample, col = 2,lwd=2,lty=2)

qqnorm(sample.dcauchy,main = 'QQ-Norm of Cauchy Dist.',xlab =
paste0('Skewness = ',
       round(as.numeric(sk_cauchy[1]),4),'\n','Kurtosis = ',
       round(as.numeric(sk_cauchy[2]),4)))
qqline(sample.dcauchy, col = 2,lwd=2,lty=2)

qqnorm(sample.dweibull,main = 'QQ-Norm of Weibull Dist.',xlab =
paste0('Skewness = ',
       round(as.numeric(sk_weibull[1]),4),'\n','Kurtosis = ',
       round(as.numeric(sk_cauchy[2]),4)))
qqline(sample.dweibull, col = 2,lwd=2,lty=2)

qqnorm(sample.dexp,main = 'QQ-Norm of Double Exp. Dist.',xlab =
paste0('Skewness = ',
       round(as.numeric(sk_dexp[1]),4),'\n','Kurtosis = ',
```

```
        round(as.numeric(sk_dexp[2]),4)))
qqline(sample.dexp, col = 2,lwd=2,lty=2)
```



Part d

Repeat Part b, but with box-whisker plots.

```
par(mfrow=c(2,2))

boxplot(norm.sample,main = 'Box-Whisker of Normal Dist.',xlab =
paste0('Skewness = ',
        round(as.numeric(sk_normdist[1]),4),'\n','Kurtosis = ',
        round(as.numeric(sk_normdist[2]),4)))

boxplot(sample.dcauchy,main = 'Box-Whisker of Cauchy Dist.',xlab =
paste0('Skewness = ',
        round(as.numeric(sk_cauchy[1]),4),'\n','Kurtosis = ',
        round(as.numeric(sk_cauchy[2]),4)))

boxplot(sample.dweibull,main = 'Box-Whisker of Weibull Dist.',xlab =
paste0('Skewness = ',
        round(as.numeric(sk_weibull[1]),4),'\n','Kurtosis = ',
        round(as.numeric(sk_weibull[2]),4)))

boxplot(sample.dexp,main = 'Box-Whisker of Double Exp. Dist.',xlab =
paste0('Skewness = ',
        round(as.numeric(sk_dexp[1]),4),'\n','Kurtosis = ',
        round(as.numeric(sk_dexp[2]),4)))
```

Hints for SAS. If you create the samples in IML, use

```
Normal = j(1, 10000, .);
call randgen(Normal, "NORMAL", 0, `);
```

You can generate samples in the data step using

```
do i = 1 to 10000;
  Normal = rand('NORMAL',0,1);
  output;
end;
```

RAND doesn't provide a Laplace option, but you can create samples from this distribution by

```
rand('EXPONENTIAL')-rand('EXPONENTIAL');
```

To group multiple plots, use

```
ods graphics / width=8cm height=8cm;
ods layout gridded columns=2;
ods region;
 ... first plot

ods region;
 ... second plot

ods layout end;
```

You might need to include

```
ods graphics off;

ods graphics on;
ODS GRAPHICS / reset=All;
```

to return the SAS graphics output to normal.

Exercise 3.

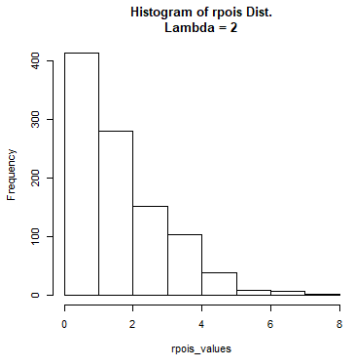
We will create a series of graphs illustrating how the Poisson distribution approaches the normal distribution with large λ. We will iterate over a sequence of `lambda`, from 2 to 64, doubling `lambda` each time. For each 'lambda' draw 1000 samples from the Poisson distribution.

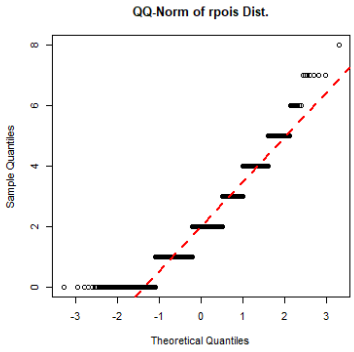
Calculate the skewness of each set of samples, and produce histograms, QQ-norm and box-whisker plots. You can use `par(mfrow=c(1,3))` to display all three for one `lambda` in one line. Add `lambda=##` to the title of the histogram, and `skewness=##` to the title of the box-whisker plot.

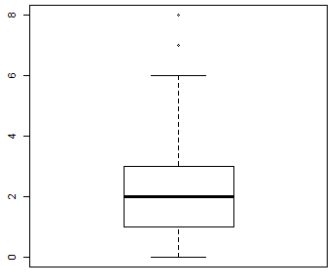
Part b.

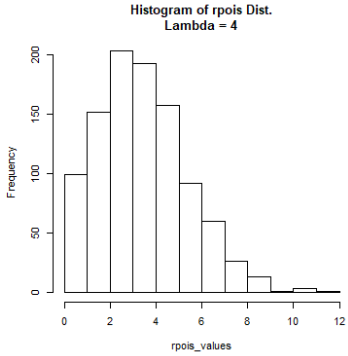
Remember that `lambda` represents the mean of a discrete (counting) variable. At what size mean is Poisson data no longer skewed, relative to normally distributed data? You might run this 2 or 3 times, with different seeds; this number varies in my experience.

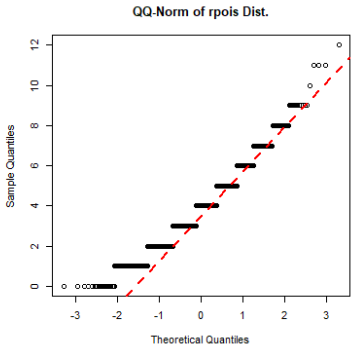
```
set.seed(54321)
mu <- 2

while(mu <= 64){
  rpois_values <- rpois(1000, lambda = mu)
  rpois_skew <- round(as.numeric(skew_kurt(rpois_values)[1]),2)

  par(mfrow=c(1,3))

  hist(rpois_values,main = paste0('Histogram of rpois Dist.','\n',
                                  'Lambda = ', mu))
  qqnorm(rpois_values,main = 'QQ-Norm of rpois Dist.')
  qqline(rpois_values, col = 2,lwd=2,lty=2)
  boxplot(rpois_values,main = paste0('Box-Whisker of rpois Dist.','\n',
                                     'Skewness = ',rpois_skew))

  mu <- mu * 2
}
```

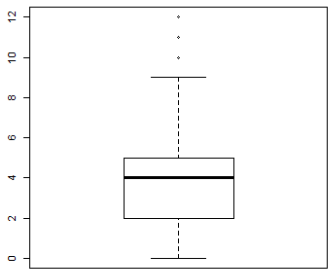

Histogram of rpois Dist.
Lambda = 2
Frequency
rpois_values

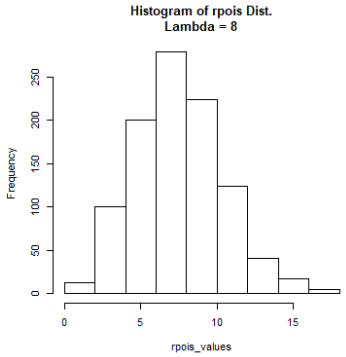

QQ-Norm of rpois Dist.
Sample Quantiles
Theoretical Quantiles

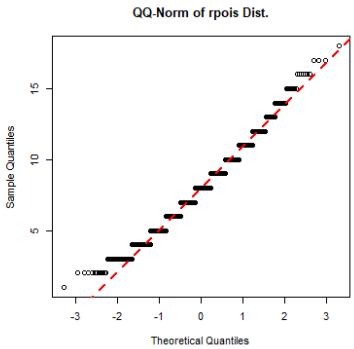

Box-Whisker of rpois Dist.
Skewness = 0.77

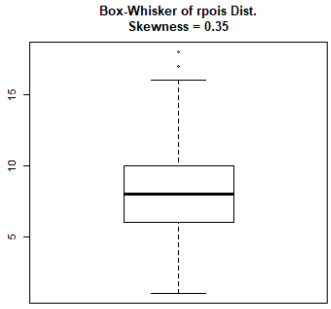

Histogram of rpois Dist.
Lambda = 4
Frequency
rpois_values

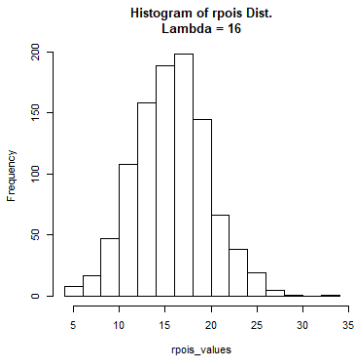

QQ-Norm of rpois Dist.
Sample Quantiles
Theoretical Quantiles

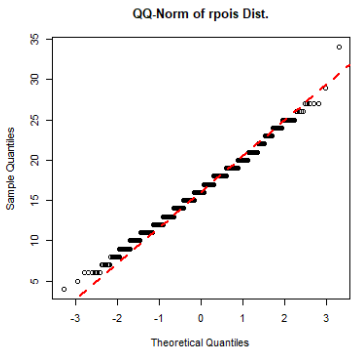

Box-Whisker of rpois Dist.
Skewness = 0.51

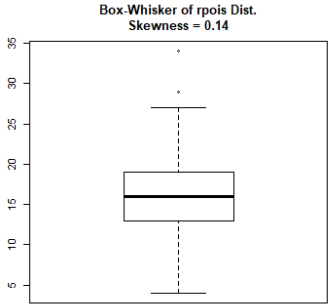

Histogram of rpois Dist.
Lambda = 8
Frequency
rpois_values

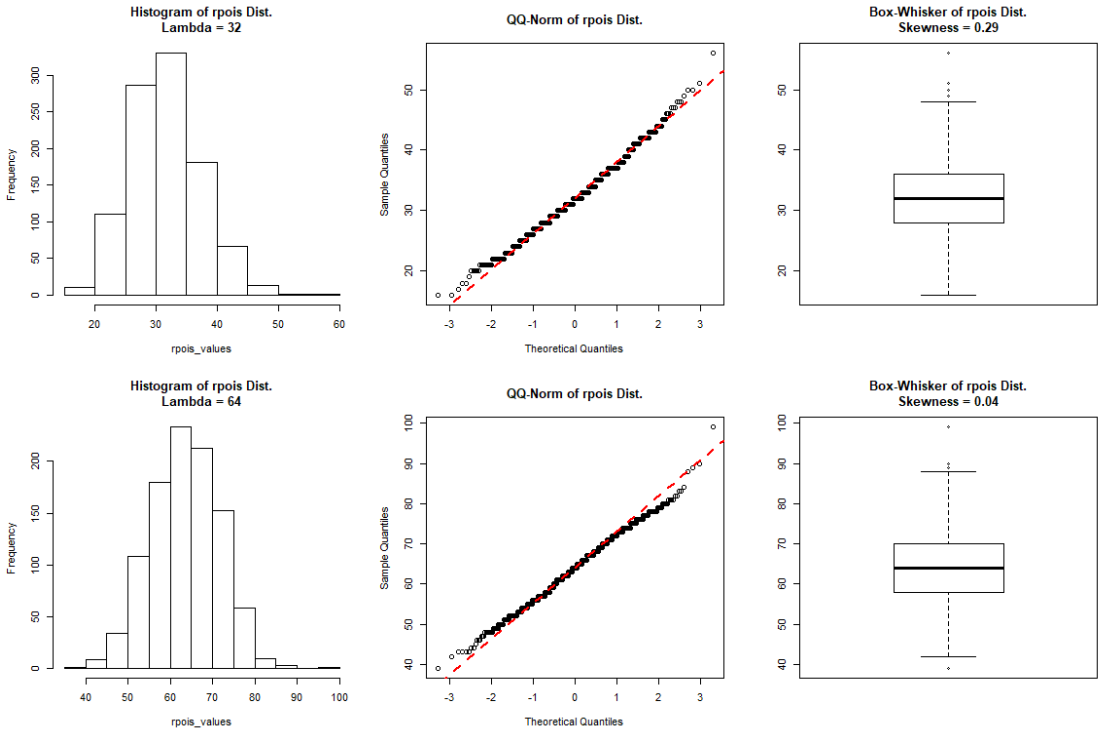

QQ-Norm of rpois Dist.
Sample Quantiles
Theoretical Quantiles


Box-Whisker of rpois Dist.
Skewness = 0.35


Histogram of rpois Dist.
Lambda = 16
Frequency
rpois_values


QQ-Norm of rpois Dist.
Sample Quantiles
Theoretical Quantiles


Box-Whisker of rpois Dist.
Skewness = 0.14



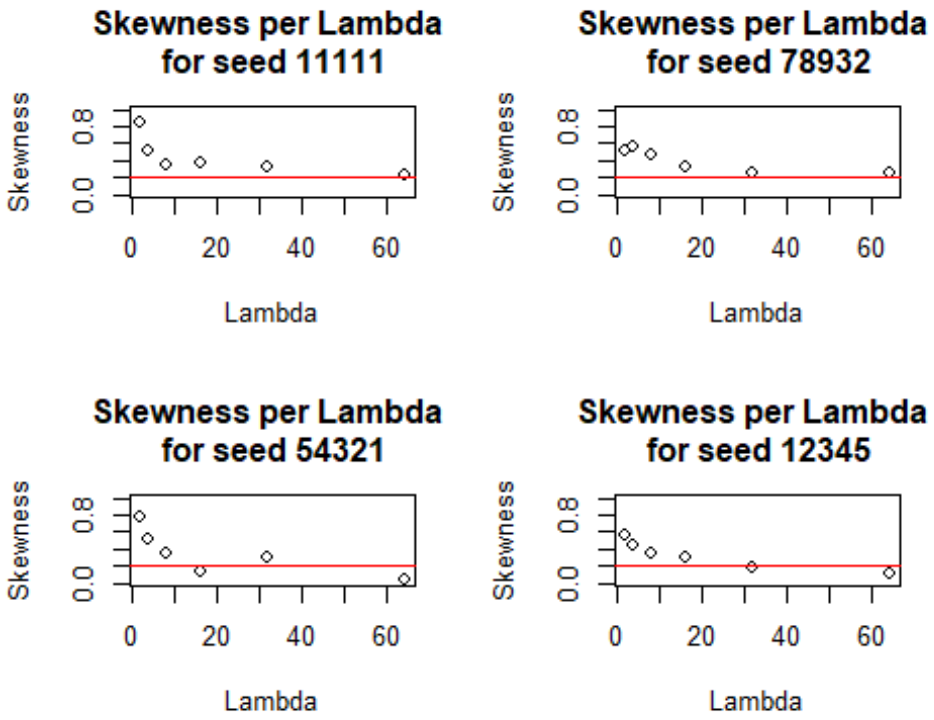
If you do this in SAS, create a data table with data columns each representing a different μ. You can see combined histogram, box-whisker and QQ-norm, for all columns, by calling

```
proc univariate data=Distributions plot;
run;
```

At what μ is skewness of the Poisson distribution small enough to be considered normal?

```
# install.packages('moments')
suppressMessages(suppressWarnings(library(moments)))
seeds <- c(11111,78932,54321,12345)
skew.dat <- data.frame(seed=numeric(6),lambda=numeric(6),
                       skewness=numeric(6))
par(mfrow=c(2,2))
for(i in 1:length(seeds)){
  k = 1
  set.seed(seeds[i])
  mu <- 2
  while(mu <= 64){
    rpois_values <- rpois(1000, lambda = mu)
    x <- skewness(rpois_values)
    skew.dat[k,1] <- seeds[i]
    skew.dat[k,2] <- mu
    skew.dat[k,3] <- x
    k = k + 1
    mu = mu * 2
  }
  plot(skew.dat[,3]~skew.dat[,2],xlab='Lambda',ylab = 'Skewness',
```

```
       main = paste0('Skewness per Lambda \n for seed ',seeds[i]),
       ylim=c(0,1))
  abline(h=0.2,col = 'red')
}
```



Comment - Based on an assumption that a sufficiently normal sample has a skewness around 0.2, I ran skewness tests for 4 different seeds. The only lambda value where all 4 seeds had a skewness close to 0.2 was 64.However, generally speaking lambda = 32 was generally close as well.

Exercise 4

Part a

Write a function that accepts a vector `vec`, a vector of integers, a main axis label and an x axis label. This function should 1. iterate over each element *i* in the vector of integers 2. produce a histogram for `vec` setting the number of bins in the histogram to *i* 3. label main and x-axis with the specified parameters. 4. label the y-axis to read `Frequency, bins =` and the number of bins.

Hint: You can simplify this function by using the parameter `...` - see `?plot` or `?hist`

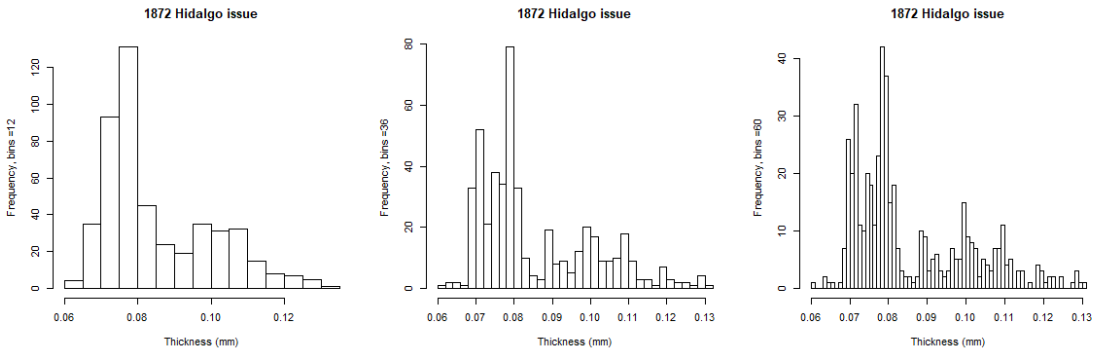
```
plot.histograms <- function(dat,vec,main,xlab){ # begin function
  par(mfrow=c(1,length(vec)))
  for(i in 1:length(vec)){ # begin for loop
```

```
    hist(dat, breaks = vec[i], main = main, xlab= xlab,
                ylab = paste0('Frequency, bins =', vec[i]))
  } # end for loop
} # end function


hidalgo.dat <- read.table('hidalgo.dat',header = FALSE)
```

Part b

Test your function with the `hidalgo` data set (see below), using bin numbers 12, 36, and 60. You should be able to call your function with something like

```
plot.histograms(hidalgo.dat[,1],c(12,36,60), main="1872 Hidalgo issue",xlab=
"Thickness (mm)")
```



to plot three different histograms of the `hidalgo` data set.

If you do this in SAS, write a macro that accepts a table name, a column name, a list of integers, a main axis label and an x axis label. This macro should scan over each element in the list of integers and produce a histogram for each integer value, setting the bin count to the element in the input list, and labeling main and x-axis with the specified parameters. You should label the y-axis to read `Frequency, bins =` and the number of bins.

Test your macro with the `hidalgo` data set (see below), using bin numbers 12, 36, and 60. You should be able to call your macro with something like

```
%plot_histograms(hidalgo, y, 12 36 60, main="1872 Hidalgo issue",
xlabel="Thickness (mm)");
```

to plot three different histograms of the `hidalgo` data set.

Hint: Assume `12 36 60` resolve to a single macro parameter and use `%scan`. Your macro definition can look something like

```
%macro plot_histograms(table_name, column_name, number_of_bins, main="Main",
xlabel="X Label")
```

Data

The `hidalgo` data set is in the file `hidalgo.dat` These data consist of paper thickness measurements of stamps from the 1872 Hidalgo issue of Mexico. This data set is commonly used to illustrate methods of determining the number of components in a mixture (in this case, different batches of paper). See https://www.jstor.org/stable/2290118, https://books.google.com/books?id=1CuznRORa3EC&lpg=PA95&pg=PA94#v=onepage&q&f=false and https://books.google.com/books?id=c2_fAox0DQoC&pg=PA180&lpg=PA180&f=false .

Some analysis suggest there are three different mixtures of paper used to produce the 1872 Hidalgo issue; other analysis suggest seven. Why do you think there might be disagreement about the number of mixtures?

Comment - I would speculate that the variability of paper thickness in 1872 was fairly high since there wasn't as many controls in thickness percision for each sheet. However, thickness below 0.08 mm was the highest frequency.

Exercise 5.

We've been working with data from Wansink and Payne, Table 1:

Reproducing part of Wansink Table 1

Measure	1936	1946	1951	1963	1975	1997	2006
calories per recipe (SD)	2123.8 (1050.0)	2122.3 (1002.3)	2089.9 (1009.6)	2250.0 (1078.6)	2234.2 (1089.2)	2249.6 (1094.8)	3051.9 (1496.2)
calories per serving (SD)	268.1 (124.8)	271.1 (124.2)	280.9 (116.2)	294.7 (117.7)	285.6 (118.3)	288.6 (122.0)	**384.4** (168.3)
servings per recipe (SD)	12.9 (13.3)	12.9 (13.3)	13.0 (14.5)	12.7 (14.6)	12.4 (14.3)	12.4 (14.3)	12.7 (13.0)

However, in Homework 2, we also considered the value given in the text

The resulting increase of 168.8 calories (from 268.1 calories … to **436.9** calories …) represents a 63.0% increase … in calories per serving.

There is a discrepancy between two values reported for calories per serving, 2006. We will use graphs to attempt to determine which value is most consistent.

First, consider the relationship between Calories per Serving and Calories per Recipe:

```
Calories per Serving = Calories per Recipe / Servings per Recipe
```

Since `Servings per Recipe` is effectively constant over time (12.4-13.0), we can assume the relationship between `Calories per Serving` and `Calories per Recipe` is linear,

$$\text{Calories per Serving} = \beta_0 + \beta_1 \times \text{Calories per Recipe}$$

with Servings per Recipe $= 1/\beta_1$

We will fit a linear model, with `Calories per Recipe` as the independent variable against two sets of values for `Calories per Serving`, such that

- Assumption 1. The value in the table (384.4) is correct.
- Assumption 2. The value in the text (436.9) is correct.

We use the data:

```
Assumptions.dat <- data.frame(
  CaloriesPerRecipe = c(2123.8, 2122.3, 2089.9, 2250.0, 2234.2, 2249.6,
3051.9),
  Assumption1 = c(268.1, 271.1, 280.9, 294.7, 285.6, 288.6, 384.4),
  Assumption2 = c(268.1, 271.1, 280.9, 294.7, 285.6, 288.6, 436.9))
```

and fit linear models

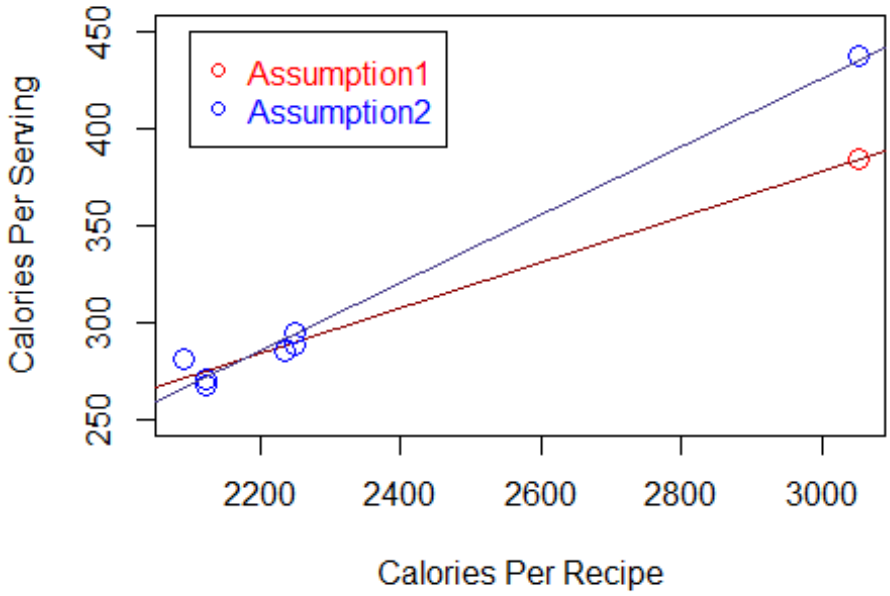
```
Assumption1.lm <- lm(Assumption1 ~ CaloriesPerRecipe,data=Assumptions.dat)
Assumption2.lm <- lm(Assumption2 ~ CaloriesPerRecipe,data=Assumptions.dat)
summary(Assumption1.lm)

##
## Call:
## lm(formula = Assumption1 ~ CaloriesPerRecipe, data = Assumptions.dat)
##
## Residuals:
##       1       2       3       4       5       6       7
## -7.0238 -3.8475  9.7610  4.7417 -2.5010 -1.3112  0.1808
##
## Coefficients:
##                    Estimate Std. Error t value Pr(>|t|)
## (Intercept)       25.477429  17.351550   1.468    0.202
## CaloriesPerRecipe  0.117547   0.007466  15.745 1.88e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6.163 on 5 degrees of freedom
## Multiple R-squared:  0.9802, Adjusted R-squared:  0.9763
## F-statistic: 247.9 on 1 and 5 DF,  p-value: 1.879e-05

summary(Assumption2.lm)

##
## Call:
## lm(formula = Assumption2 ~ CaloriesPerRecipe, data = Assumptions.dat)
##
```

```
## Residuals:
##       1       2       3       4       5       6       7
## -4.1798 -0.9169 14.5608  0.3051 -6.0261 -5.7248  1.9817
##
## Coefficients:
##                     Estimate Std. Error t value Pr(>|t|)
## (Intercept)       -99.891018  21.933161  -4.554  0.00609 **
## CaloriesPerRecipe   0.175238   0.009437  18.569 8.34e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 7.79 on 5 degrees of freedom
## Multiple R-squared:  0.9857, Adjusted R-squared:  0.9828
## F-statistic: 344.8 on 1 and 5 DF,  p-value: 8.336e-06
```

Part a.

Plot the regression. Use points to plot `Assumption1` vs `CaloriesPerRecipe`, and `Assumption2` vs `CaloriesPerRecipe`, on the same graph. Add lines (i.e. `abline`) to show the fit from the regression. Use different colors for the two assumptions. Which of the two lines appears to best explain the data?

```
par(mfrow=c(1,1))
plot(Assumptions.dat$Assumption1~Assumptions.dat$CaloriesPerRecipe, cex =
1.5, type = 'p', col = 'red',
     ylim = c(250,450), ylab = 'Calories Per Serving',
     xlab = 'Calories Per Recipe')
abline(Assumption1.lm, col = 'darkred')
points(Assumptions.dat$Assumption2~Assumptions.dat$CaloriesPerRecipe, cex =
1.5, col = 'blue')
abline(Assumption2.lm, col = 'darkslateblue')
legend(2100,450,legend = c('Assumption1','Assumption2'),col=c('red','blue'),
       text.col=c('red','blue'),pch=c(1,1))
```



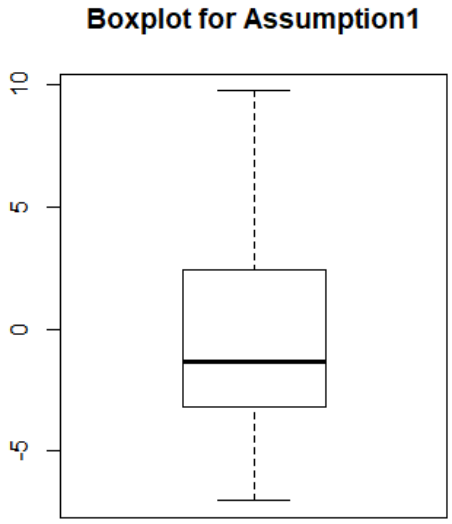
Comment - Based on the plot, it is difficult to determine which assumption appears to best explain the data. Assumption 1 data points appears to be marginally closer to the linear trend. However, more analysis would be needed to determine the most likely assumption.

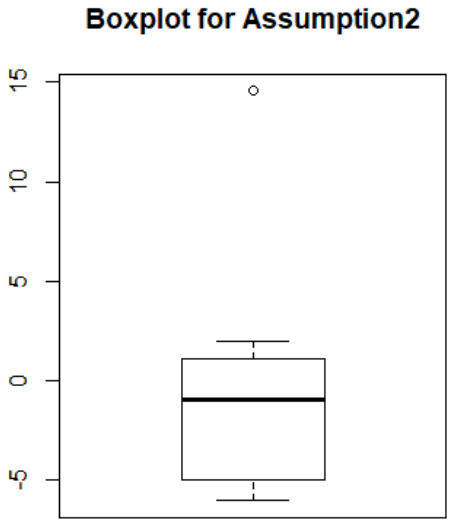
Part b.

Produce diagnostic plots plots of the residuals from both linear models (in R, use `residuals(Assumption1.lm)`). qqnorm or box-whisker plots will probably be the most effective; there are too few points for a histogram.

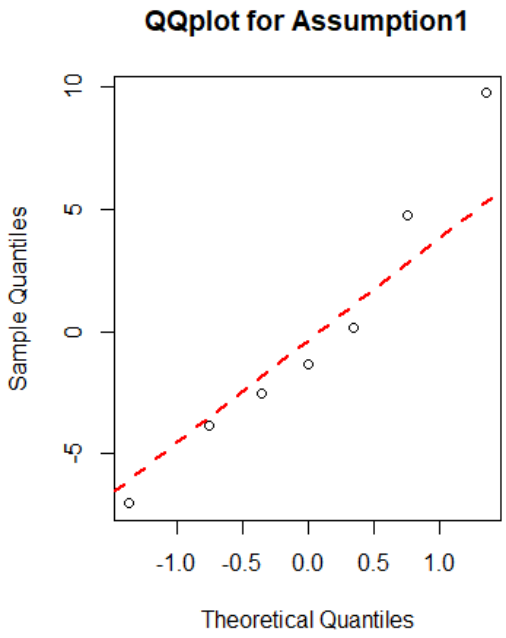
Use the code below to place two plots, side by side. You can produce more than one pair of plots, if you wise.

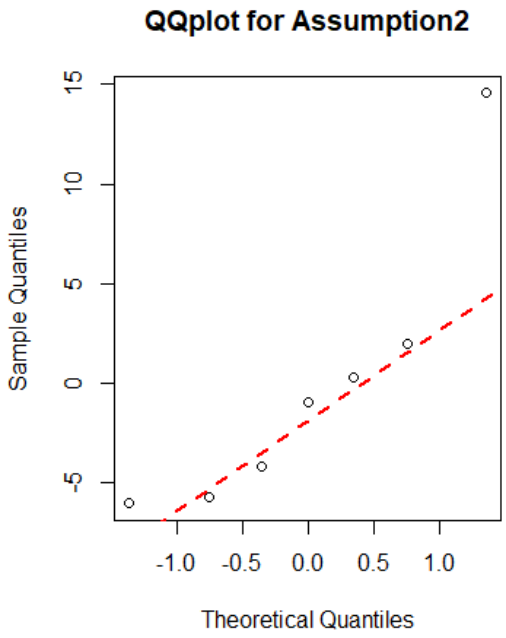
```
par(mfrow=c(1,2))

boxplot(residuals(Assumption1.lm), main = "Boxplot for Assumption1")
boxplot(residuals(Assumption2.lm), main = "Boxplot for Assumption2")
```






```r
par(mfrow=c(1,2))

qqnorm(residuals(Assumption1.lm), main = "QQplot for Assumption1" )
qqline(residuals(Assumption1.lm),col = 2,lwd=2,lty=2)

qqnorm(residuals(Assumption2.lm), main = "QQplot for Assumption2" )
qqline(residuals(Assumption2.lm),col = 2,lwd=2,lty=2)
```





From these plots, which assumption is most likely correct? That is, which assumption produces a linear model that least violates assumptions of normality of the residual errors? Which assumption produces outliers in the residuals?

I've included similar data and linear models for SAS in the SAS template. If you choose SAS, you will need to modify the PROC GLM code to produce the appropriate diagnostic plots.

Comment - Assumption 1 least violates assumptions of normality of the residual errors and appears to be more linear. Assumption 2 produces the outlier in the residuals based on the box whisker plots and is apparent in the QQ-plot as well.