18.S096 Problem Set 3 Spring 2018
Due Date: 3/9/2018
Where: On Stellar, prior to 11:59pm

Collaboration on homework is encouraged, but you will benefit from independent effort to solve the problems before discussing them with other people. **You must write your solution in your own words. List all your collaborators.**

1. **MLE for Truncated $Poisson(\lambda)$ Distribution**

   Suppose sample observations $X_1, \ldots, X_n$ from a $Poisson(\lambda)$ distribution are truncated so that the observations are $Y_1, \ldots, Y_n$ where

$$Y_i = \begin{cases} 0, & if \quad X_i = 0 \\ 1, & if \quad X_i > 0 \end{cases}$$

   1(a) Derive the MLE for λ given the truncated sample $Y_1, \ldots, Y_n$.

   Solution:

$$\begin{aligned} \ell(\lambda) &= \textstyle\prod_{i=1}^{n} (e^{-\lambda})^{1-y_i} (1 - e^{-\lambda})^{y_i} \\ &= (1-\theta)^{n - \sum_1^n y_i} (\theta)^{\sum_1^n y_i} \\ &= \ell_*(\theta) \\ &\quad where \;\; \theta = 1 - e^{-\lambda} \end{aligned}$$

   Solving the first-order condition

$$\tfrac{\partial \ell(\theta)}{\partial \theta} = 0,$$

   the maximum of $\ell_*(\theta)$ is achieved at

$$\hat{\theta} = \tfrac{1}{n} \textstyle\sum_1^n y_i = \overline{Y}.$$

   Solving for λ in terms of θ gives

$$\hat{\lambda} = -\ln(1 - \overline{Y}).$$

   This uses the fact that if $\hat{\theta}$ is the MLE of θ, then $g(\hat{\theta})$ is the MLE of $g(\theta)$.

   Note that the MLE of λ is $+\infty$ when $\overline{Y} = 1$. (Also, when $\overline{Y} = 0$, then the MLE is 0.) This possibility complicates evaluation of the statistical precision of $\hat{\lambda}_{MLE}$. The sampling distribution for the MLE has an infinite value with positive probability, which will make both the expectation (mean) and variance of the sampling distribution of the MLE infinite valued.

   Statistical inference of this model would apply the asymptotic distribution theory of MLEs, i.e., in large samples the MLE is well-approximated by a Gaussian distribution. For this distribution:

$$mean = \lambda,$$

   the true parameter value, and

$$variance = [n\mathcal{I}(\lambda)]^{-1},$$

the inverse of the *Fisher Information* for a sample of size n. The *Fisher Information* for a single observation is:

$$\mathcal{I}(\lambda) = E[(\frac{\partial log f(x|\lambda)}{\partial \lambda})^2] = E[-\frac{\partial^2 log f(x|\lambda)}{\partial \lambda^2}]]$$

The *Fisher Information* for the sample of n realizations can be estimated by

$$Estimate\ of\ n\mathcal{I}(\lambda) = [\nabla \ell(\hat{\lambda})]^2.$$

This estimate is called the *observed* or *empirical Fisher Information*.

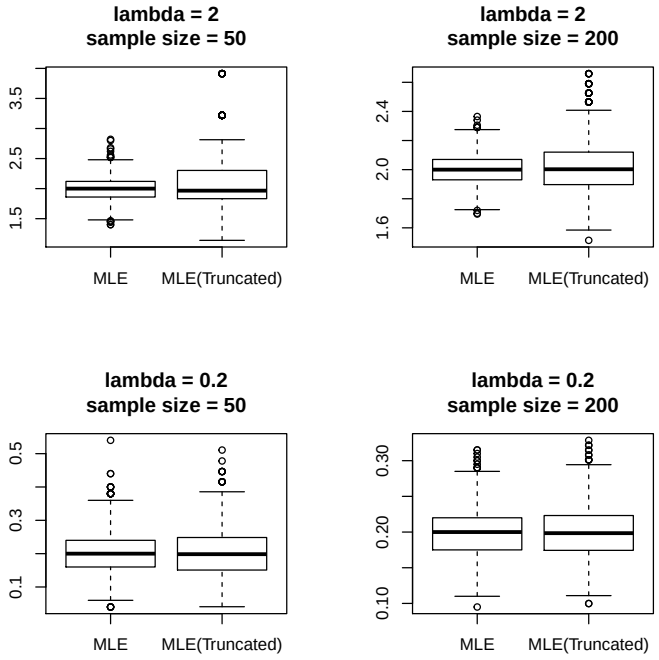
1(b) Using the R function *rpois*() to generate random Poisson variates, conduct a Monte Carlo simulation comparing the sampling distribution of the Truncated MLE to the sampling distribution of the regular MLE.

- Simulate sampling distributions for two cases of $\lambda : \lambda = 2$, and $\lambda = 1/5$.
- Consider two cases of the sample size: $n = 50$ and $n = 200$.

For each case of the simulation (choice of lambda and choice of sample size) compare these distributions by constructing a parallel boxplot and computing sample means/standard deviations of the distributions.

```
> par(mfcol=c(2,2))
> #n.samplesize=50
> n.trials=2000
> # Case 1
> list.lambda=c(2,1/5)
> list.samplesize=c(50,200)
> for (n.samplesize in list.samplesize){
+ for (j.lambda in c(1:length(list.lambda))){
+   #j.lambda<-1
+   lambda=list.lambda[j.lambda]
+
+   table.estimates<-matrix(NA,nrow=n.trials, ncol=2)
+   dimnames(table.estimates)[[2]]<-c("MLE", "MLE(Truncated)")
+ set.seed(1)
+ for (j in 1:n.trials){
+   x=rpois(n.samplesize, lambda)
+   x.mle=mean(x)
+   y=ifelse(x>0,0*x +1,0*x)
+   y.mle= - log(1-mean(y))
+ table.estimates[j,]<-c(x.mle, y.mle)
+ }
+ table.estimates.summary<-cbind(
+   mean= apply(table.estimates,2,mean),
+   sd=sqrt(apply(table.estimates,2,var)))
+ table.estimates.finite<-ifelse(is.infinite(table.estimates),NA, table.estimates)
+
+ table.estimates.finite.summary<-cbind(
+   mean=apply(table.estimates.finite,2,mean,na.rm=TRUE),
```

```
+   sd=sqrt(apply(table.estimates.finite,2,var,na.rm=TRUE)))
+ boxplot(table.estimates, main=paste(
+   c("lambda = ", as.character(lambda),"\nsample size = ",n.samplesize),
+   collapse=""))
+ assign(paste(c("table.estimates.case",
+                round(lambda,digits=2),n.samplesize),collapse="."),
+              table.estimates)
+ assign(paste(c("table.estimates.summary",
+                round(lambda,digits=2),n.samplesize),collapse="."),
+        table.estimates.summary)
+ assign(paste(c("table.estimates.finite.summary",
+                round(lambda,digits=2),
+                n.samplesize),collapse="."),
+        table.estimates.finite.summary)
+ }}
```



Note: infinite values occurred with boxplot 2. R gave the following warning message

```
Warning message:
In bplt(at[i], wid = width[i], stats = z$stats[, i], out = z$out[z$group ==  :
  Outlier (Inf) in boxplot 2 is not drawn
```

```
> # Because estimates can be infinite
```

```
> # Summary statistics computed with sub-samples
> #     excluding infinite estimates
> #
> # First case is lambda=2.
> print(table.estimates.summary.2.50)

                  mean       sd
MLE            1.99989 0.196904
MLE(Truncated)     Inf      NaN

> print(table.estimates.finite.summary.2.50)

                   mean        sd
MLE            1.999890 0.1969040
MLE(Truncated) 2.055189 0.4038107

> # Note that truncated mle has higher sd
>
> # For the larger sample size (n=200):
> print(table.estimates.summary.2.200)

                   mean        sd
MLE            2.000905 0.1004182
MLE(Truncated) 2.015995 0.1805552

> print(table.estimates.finite.summary.2.200)

                   mean        sd
MLE            2.000905 0.1004182
MLE(Truncated) 2.015995 0.1805552

> # There were no infinite values of the estimates
> #   with the larger sample size
>
> # Note that the larger sample size (by a factor of 4)
> # had sd's that are about 1/2 those of the smaller sample
> # size.  This is consistent with asymptotic distribution
> # theory of mle's.
>
> # Second case is lambda=0.2.
> print(table.estimates.summary.0.2.50)

                  mean         sd
MLE            0.20064 0.06318534
MLE(Truncated) 0.20238 0.06700700

> print(table.estimates.finite.summary.0.2.50)

                  mean         sd
MLE            0.20064 0.06318534
MLE(Truncated) 0.20238 0.06700700
```

```
> # Note that truncated mle has higher sd
>
> # For the larger sample size (n=200):
> print(table.estimates.summary.0.2.200)

                    mean         sd
MLE            0.2001450 0.03250819
MLE(Truncated) 0.2002998 0.03419406

> print(table.estimates.finite.summary.0.2.200)

                    mean         sd
MLE            0.2001450 0.03250819
MLE(Truncated) 0.2002998 0.03419406

> # Note: no infinite values in either simulated sample
>
> # Comparing across sample sizes, the standard errors (sd)
> # of the estimates decreases by a factor of about sqrt(4)=2
> print(table.estimates.summary.0.2.50)

                   mean         sd
MLE            0.20064 0.06318534
MLE(Truncated) 0.20238 0.06700700

> print(table.estimates.summary.0.2.200)

                    mean         sd
MLE            0.2001450 0.03250819
MLE(Truncated) 0.2002998 0.03419406
```

- The sampling distribution of the full-sample MLE has less dispersion than that of the truncated-sample MLE – this is apparent in the parallel boxplots above. The truncated sample has less information than the full sample so it makes sense that the MLEs would be less precise.
- The sampling distributions of the MLEs are more different for the larger λ values (when comparing for the same sample size). The information content of the truncated values decreases the larger λ is (above 1).

1(c) For the simulation in part (b), answer the following questions:

- How does the relative efficiency (variance ratio) of the two estimates depend on λ?
- How does the absolute efficiency (variances) of the two estimates depend on λ and the sample size?
- Is there an issue with the truncated MLE ever being infinite. If so, how should this be taken in account with the comparisons?

Solution:

- The relative efficiency of the truncated to the full-sample MLE decreases as λ increases.
- As the sample size increases, the MLEs have standard errors (estimated standard deviations) which decrease with the square root of the sample size. The standard errors for $n = 200$ should be about half the size of those for $n = 50$,; this is consistent with the asymptotic distribution theory of maximum likelihood esimates. The relative efficiency of the full-sample versus truncated MLEs should stay about the same for all sample sizes.
- The absolute efficiency as measured by the variances of the estimates decreases increases with the sample size. It is also proportional to λ for the full-sample MLE.

  For the truncated MLE, a delta argument is required:
  Per A. Konstantinov:
  $\hat{p} = 1 - e^{-\hat{\lambda}} \Longrightarrow \hat{\lambda} = -log(1 - \ p)$
  By the CLT, $\sqrt{n}(\hat{p} - p) \to N(0, p(1-p))$
  Now $f(x) = 1 - log(1-x)$ and $f'(x) = \frac{df(x)}{dx} = \frac{1}{1-x}$.
  So by the delta method:
  $\sqrt{n}(f(\hat{p}) - f(p)) \to N(0, p(1-p) \cdot [f'(p)]^2)$
  $\Longrightarrow \sqrt{n}(\hat{\lambda} - \lambda) \to N(0, \frac{p}{1-p})$
  Note that $\frac{p}{1-p} = \frac{1-e^{-\lambda}}{e^{-\lambda}=e^{\lambda}-1}$
  Thus $\sqrt{n}(\hat{\lambda} - \lambda) \to N(0, e^{\lambda} - 1)$.
  Since $e^{+\lambda} - 1 > \lambda$, the truncated MLE has asymptotic variance greater than that of the full-sample MLE.
  (The delta argument is covered in 18.650 and 18.655.)
- There an issue with the truncated MLE being infinite. When λ is very large, all the truncated values may be 1, so the estimate of λ with the truncated data would be infinite.
  When comparing estimates, it could be relevant to determine the proportion of samples in which one expects infinite/undetermined estimates to result; if this proportion approaches zero, then the asymptotic Gaussian distribution for the mle would apply.

2. The R function $fitdistr()$ in the R package $MASS$ fits univariate distributions by maximum likelihood. As detailed in $help(fitdistr)$, the syntax is:

   ```
   fitdistr(x, densfun, start, ...)
   ```

   with Arguments

   - x: A numeric vector of length at least one containing only finite values.

- *densfun*: Either a character string or a function returning a density evaluated at its first argument. Distributions "beta", "cauchy", "chi-squared", "exponential", "f", "gamma", "geometric", "log-normal", "lognormal", "logistic", "negative binomial", "normal", "Poisson", "t" and "weibull" are recognised, case being ignored.
- *start*: A named list giving the parameters to be optimized with initial values. This can be omitted for some of the named distributions and must be for others (see Details).

The output *Value* of the function is an object of class "fitdistr", a list with four components:

- *estimate* : the parameter estimates,
- *sd*: the estimated standard errors,
- *vcov*: the estimated variance-covariance matrix, and
- *loglik*: the log-likelihood.

The following R code simulates a sample from the Gamma distribution and applies *fitdistr*() to estimate the distribution parameters:

```
> #
> library(MASS)
> set.seed(1)
> samplesize=100
> x=rgamma(samplesize,shape=3, rate=1)
> fit_gamma_mle<-fitdistr(x,densfun="gamma")
> print(fit_gamma_mle)

     shape       rate
  3.7673500   1.2919621
 (0.5109475) (0.1874392)

> names(fit_gamma_mle)

[1] "estimate" "sd"       "vcov"     "loglik"   "n"

> fit_gamma_mle$estimate

   shape     rate
3.767350 1.291962

> fit_gamma_mle$sd

    shape      rate
0.5109475 0.1874392

> fit_gamma_mle$vcov
```

```
          shape       rate
shape 0.26106737 0.08952941
rate  0.08952941 0.03513346
```

```
> fit_gamma_mle$loglik
```

```
[1] -173.1452
```

```
> fit_gamma_mle$n
```

```
[1] 100
```

2(a) Generate 4 samples from a $Gamma(shape = 3, rate = 2)$ distribution with sample sizes: 100, 400, 800, 1600. Apply $fitdistr()$ to compute the mles and standard errors for each sample

```
> # Generate random samples from a
> #     Gamma(shape=3,rate=2) distribution
> set.seed(1)
> list.samplesize<-c(100,400,800,1600)
> for (samplesize in list.samplesize){
+   assign(paste("sample.gamma",samplesize,sep="."),
+          rgamma(samplesize, shape=3,rate=2))}
> mlefit.sample.gamma.100<-fitdistr(sample.gamma.100,
+                                   densfun="gamma")
> mlefit.sample.gamma.400<-fitdistr(sample.gamma.400,
+                                   densfun="gamma")
> mlefit.sample.gamma.800<-fitdistr(sample.gamma.800,
+                                   densfun="gamma")
> mlefit.sample.gamma.1600<-fitdistr(sample.gamma.1600,
+                                   densfun="gamma")
```

2(b) How does the sd of the mle's depend on the sample size? Is this consistent with maximum likelihood theory/asymptotics?

```
> mlefit.sample.gamma.100
```

```
     shape       rate
  3.7673500   2.5839242
 (0.5109480) (0.3748789)
```

```
> mlefit.sample.gamma.400
```

```
     shape       rate
  2.7979960   1.8846309
 (0.1872558) (0.1381376)
```

```
> mlefit.sample.gamma.800
```

```
     shape         rate
  2.81586162   1.82749503
 (0.13329736) (0.09469172)
```

```
> mlefit.sample.gamma.1600

     shape         rate
  3.13996742   2.07596884
 (0.10564925) (0.07574124)

>
```

The *sd* decreases with the sample size – approximately as the square-root of the sample size, which is consistent with mle theory/asymptotics.

2(c) Define the function *mydgamma*() to compute the density of a gamma distribution applying its formula:

$$f(x \mid \alpha, \beta) = \frac{1}{\Gamma(\alpha)} \beta^{\alpha} x^{\alpha-1} e^{-\beta x},\ x > 0.$$

Apply $fitdistr()$ with this user-defined density to compute the mle's for the four samples. Confirm that you obtain results consistent with part (a).

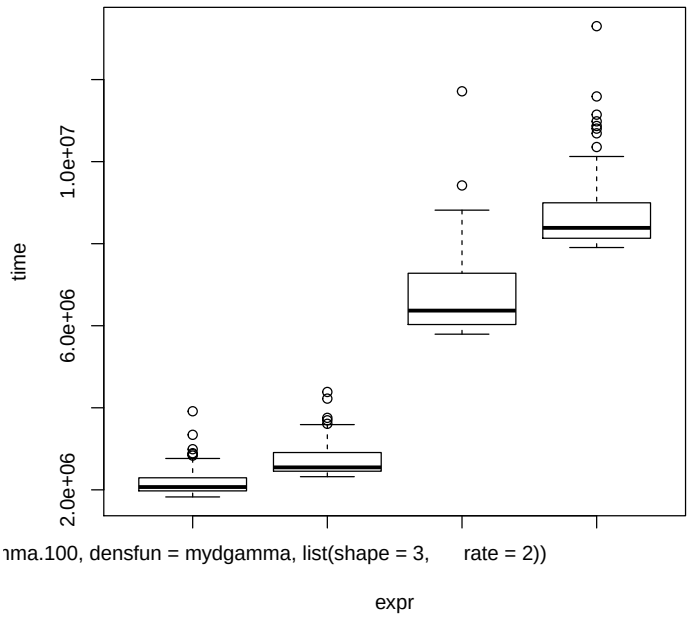
```
> mydgamma<-function(x,shape,rate){
+   dens=(1/gamma(shape))*(rate^{shape})*(x^{shape-1}) *exp(-rate*x)
+   }
> fitdistr(sample.gamma.100,densfun=mydgamma,
+          start=list(shape=3,rate=2))

     shape       rate
  3.7669840   2.5836241
 (0.5108978) (0.3748376)

> fitdistr(sample.gamma.400,densfun=mydgamma,
+          start=list(shape=3,rate=2))

     shape       rate
  2.7979921   1.8846277
 (0.1872557) (0.1381376)

> fitdistr(sample.gamma.800,densfun=mydgamma,
+          start=list(shape=3,rate=2))

     shape         rate
  2.81583605   1.82747535
 (0.13329626) (0.09469089)

> fitdistr(sample.gamma.1600,densfun=mydgamma,
+          start=list(shape=3,rate=2))

     shape         rate
  3.13996158   2.07596464
 (0.10564920) (0.07574123)

>
```

These results are consistent with those from part (a). Differences are due to numerical precision of the estimates in the optimization being only about 4 significant digits.

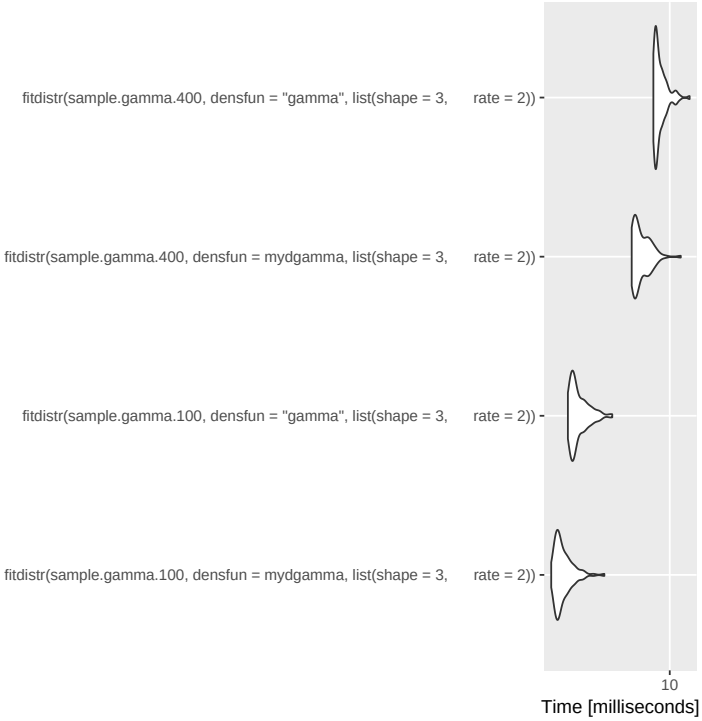
2(d) Use the R function $microbenchmark()$ in $library(microbenchmark)$ to compare the computation times of the two options

$$densfun = "gamma" \text{ and } densfun = mydgamma$$

```
> library(microbenchmark)
> compare.gammas<-microbenchmark(
+   fitdistr(sample.gamma.100, densfun = mydgamma,
+            list(shape=3,rate=2)),
+   fitdistr(sample.gamma.100, densfun = "gamma",
+            list(shape=3,rate=2)),
+   fitdistr(sample.gamma.400, densfun = mydgamma,
+            list(shape=3,rate=2)),
+   fitdistr(sample.gamma.400, densfun = "gamma",
+            list(shape=3,rate=2)))
> library(ggplot2)
> plot(compare.gammas)
```




```
> autoplot(compare.gammas)
```



Is one option generally faster on average than the other? If so, what would explain the difference?

The user-defined *mydgamma*() case is faster. This is probably due to the overhead in the function $fitdistr()$. However these results are for a PC; with the MAC the results may be the opposite!

3. **Laplace Distribution**

Let $X_1, X_2, \ldots X_n$ be a random sample from the $Laplace(\mu, b)$ distribution with density:

$$f(x \mid \mu, b) = \frac{1}{2b} e^{-\frac{|x-\mu|}{b}}, \; -\infty < x < \infty.$$

with two parameters:

μ (location)

b (scale)

3(a) Derive formulas for the method-of-moments estimates of μ and b.

$$\begin{array}{lllllll} mean & = & E[X] & = & \mu & = & \int_{-\infty}^{+\infty} x f(x \mid \mu, \sigma) dx \\ variance & = & Var[X] & = & 2 \times b^2 & = & \int_{-\infty}^{+\infty} (x-\mu)^2 f(x \mid \mu, \sigma) dx \end{array}$$

So,

$$\hat{\mu}_{MOM} = \bar{X} = \frac{1}{n} \sum_{i=1}^{n} X_i,$$

and

$$\hat{\sigma} = \sqrt{\widehat{Var}(X)/2} \text{ where}$$

$\widehat{Var}(X) = \frac{1}{n}\sum_{i=1}^{n}(X_i - \bar{X})^2$

3(b) Derive formulas for the maximum-likelihood estimates of μ and b.

The log-likelihood for the sample $x_1, \ldots, x_n$ is:

$$\begin{aligned} \ell(\mu, b) &= \sum_{i=1}^{n} log[f(x_i \mid \mu, b) \\ &= -nlog(2) - nlog(b) - \sum_{i=1}^{n} \frac{|x_i - \mu|}{b} \end{aligned}$$

For any b, th$\ell(\mu, b)$ is minimized by

$$\hat{\mu} = median(x_i).$$

The derivative of the log-likelihood with respect to μ is positive when μ is less than half the x_i and is negative when it is greater than half he x_i. When n is odd this is the median, when n is even, any value between the two middle values maximizes the likelihood.

Given $\hat{\mu} = median(x_i)$, the log likelihood is maximized by

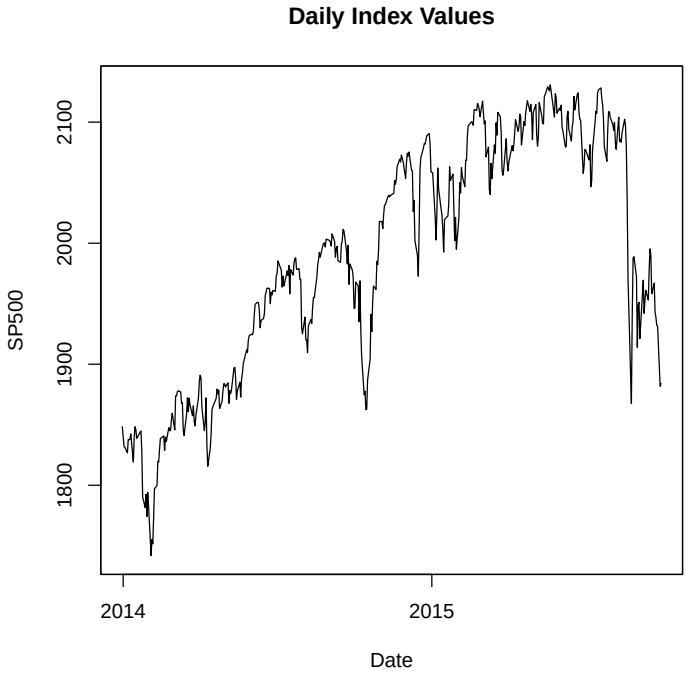
$$\hat{b} = \frac{1}{n}\sum_{i=1}^{n} |x_i - \hat{\mu}|.$$

This is proven by substituting in $\mu = \hat{\mu}$ and solving the first-order condition for $\hat{b}$.

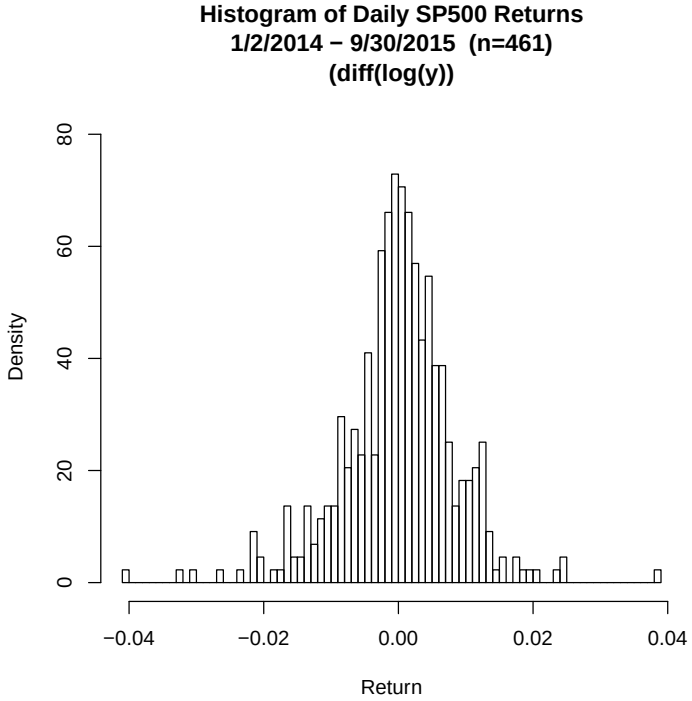
3(c) The file $SP500.csv$ has daily values of the S&P 500 stock index from 1/2/2014 to 9/30/2015.

- In R, install the package "zoo" (for time series) and use $read.zoo()$ to read the data into R.
- Plot the time series
- Compute the logarithmic daily returns and plot the histogram.

```
> # install.packages("zoo")
> library(zoo)
> SP500<-read.zoo(file="SP500.csv")
> par(mfcol=c(1,1))
> plot(SP500, main="Daily Index Values", xlab="Date")
```



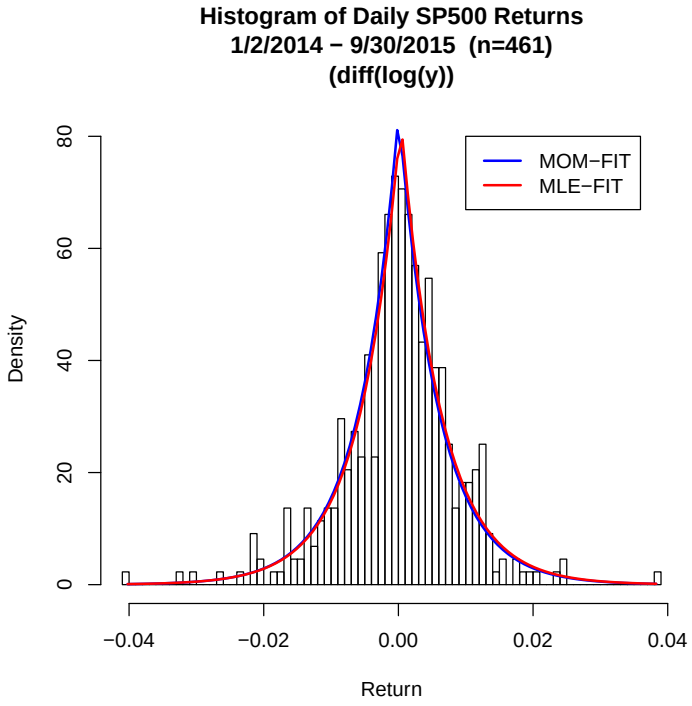

```
> y<-diff(log(SP500))
> hist( y, breaks=100,
+      ylab="Density",xlab="Return",
+      freq=FALSE,ylim=c(0,84.),
+      main=paste(c("Histogram of Daily SP500 Returns",
+                   "1/2/2014 - 9/30/2015  (n=461)",
+                   "(diff(log(y))"),collapse="\n"))
>
```



3(d) Fit the parameters of the Laplace distribution by method-of-moments and by maximum likelihood; report the estimates and draw the fitted density from each method on the histogram.

```
> # Method of moments estimates
> mu.mean=mean(y)
> sigma.laplace=sqrt( mean((y-mu.mean)^2)/2)
> ymax.laplace=0.5/sigma.laplace
> # Method-of-Moments Estimates
> location.mom=mean(y)
> scale.mom=sqrt( mean((y-location.mom)^2)/2)
> location.mle<-median(y)
> scale.mle<-mean(abs(y-location.mle))
> param.mle<-list(location=location.mle, scale=scale.mle)
> param.mom<-list(location=location.mom, scale=scale.mom)
> tab.estimates<-cbind(
+   location=c(location.mom, location.mle),
+   scale=c(scale.mom, scale.mle))
> dimnames(tab.estimates)[[1]]<-c("Method-of-Moments",
+                                 "Maximum Likelihood")
> round(tab.estimates,digits=6)
                   location    scale
Method-of-Moments   4.4e-05 0.005942
```

```
Maximum Likelihood  3.5e-04 0.006029
>
> ymax.laplace=0.5/sigma.laplace
> # Plot histogram with
> #       the mle-fitted density, and
> #       the mom-fitted density
> hist( y, breaks=100,
+      ylab="Density",xlab="Return",
+      freq=FALSE,ylim=c(0,84.),
+      main=paste(c("Histogram of Daily SP500 Returns",
+                   "1/2/2014 - 9/30/2015  (n=461)",
+                   "(diff(log(y))"),collapse="\n"))
> fcn.laplacedensity.mom<-function(x){
+   result=0.5*dexp(abs(x-location.mom),rate=1/scale.mom)}
> curve(fcn.laplacedensity.mom, from =min(y),to=max(y), add=TRUE,col='blue',lwd=2)
> fcn.laplacedensity.mle<-function(x){
+   result=0.5*dexp(abs(x-location.mle),rate=1/scale.mle)}
> curve(fcn.laplacedensity.mle, from =min(y),to=max(y), add=TRUE,col='red',lwd=2)
> legend(x=.01, y=80, legend=c("MOM-FIT","MLE-FIT"),
+ col=c("blue","red"),lwd=c(2,2))
```



3(e) Define the R function *dlaplace*() to compute the density function for a Laplace distribution

```
> dlaplace<-function(x, location=0,scale=1){
+   dx=(0.5/scale)* exp(-abs(x-location)/scale)
+ }
```

Use the R function $fitdistr()$ to fit the Laplace distribution by maximum likelihood. Comment on the consistency of the mle's from $fitdistr()$ and those computed using the exact formulas in (d).

3(f) Compare the mle and method-of-moments estimates (how big are the differences in terms of the sd's of the mle's output by $fitdistr()$).

```
> round(tab.estimates,digits=6)
```

```
                   location    scale
Method-of-Moments   4.4e-05 0.005942
Maximum Likelihood  3.5e-04 0.006029
```

```
> library(MASS)
> y.fitdistr.dlaplace<-fitdistr(y, densfun=dlaplace,
+                       start=list(location=0.,scale=1.))
> y.fitdistr.dlaplace
```

```
     location        scale
  0.0004299156   0.0060462367
 (0.0003127266) (0.0002656198)
```

```
> relativeDiff.mom<-(tab.estimates[1,]-y.fitdistr.dlaplace$estimate)/y.fitdistr.dla
> relativeDiff.mom
```

```
  location      scale
-1.2352721 -0.3941247
```

```
> # The MOM estimate for location is more than one sd
> # from the MLE while
> # the MOM estimate for scale is within 0.5 sd of the  MLE
> # Relative diference
>
> # The MOM and MLE fits can also be compared using their
> # maximized log likelihoods:
>
> loglik.mle<-sum(log(dlaplace(y,
+                              location=location.mle,
+                              scale=scale.mle)))
> loglik.mom<-sum(log(dlaplace(y,
+                              location=location.mom,
+                              scale=scale.mom)))
> loglik.mle
```

```
[1] 1500.528
```

```
> loglik.mom
```

```
[1] 1500.048
```

```
> # These are  close -- formal evaluation of how close
> # can be done later.
>
> # We can compare the mle log likelihood with that
> # from fitdistr() and see that a consistent value obtains
> library(MASS)
> y.fitdistr.dlaplace<-fitdistr(y, densfun=dlaplace,
+                       start=list(location=0.,scale=1.))
> y.fitdistr.dlaplace
     location        scale
  0.0004299156   0.0060462367
 (0.0003127266) (0.0002656198)
> y.fitdistr.dlaplace$loglik
[1] 1500.506
> # We can also compare the log likelihood for the
> # mle laplace fit to that for the mle normal fit
> y.fitdistr.normal<-fitdistr(y, densfun="normal")
> y.fitdistr.normal$loglik
[1] 1475.16
> #
> # The normal model has a much lower log likelihood
> # suggesting that it would be rejected in favor
> # of the Laplace disgtribution.
```

4. The workings of the R function *fitdistr*() can be understood by studying the function definition. Print the function definition by typing just the function name

   > fitdistr

   at the R console without any parentheses. To study/edit the function you can use the built-in object/function editor called *fix*(), i.e.,

   > fix(myfitdistr)

   However, many find *fix*() cumbersome to use when editing/revising functions. Instead, make a copy of the function *fitdistr*() by copying the function definition into a separate R script file. In this new file, edit the first line to give the function assignment a new name, i.e.,

```
myfitdistr<-function (x, densfun, start, ...)
{
  myfn <- function(parm, ...) -sum(log(dens(parm, ...)))
  ...
  (rest of function)
}
```

For convenience, such a file is included in the problem set materials.

Edit the function script file to include comments explaining the code at the following if-blocks:

4(a) if-block

if (distname == "poisson")

```
    if (distname == "poisson") {
      if (!is.null(start))
        stop(gettextf("supplying pars for the %s distribution is not supported",
                      "Poisson"), domain = NA)
# MLE for poisson is known to be sample mean:
      estimate <- mean(x)
# Standard error (sd) of sample mean is
# square root of estimate of variance of sample mean
#     Since variance of Poisson is the same as the mean
#     this is the formula for the sd
      sds <- sqrt(estimate/n)
      names(estimate) <- names(sds) <- "lambda"
# The variance/covariance matrix of estimates is a scalar
#     plug in sds^2
      vc <- matrix(sds^2, ncol = 1, nrow = 1,
        dimnames = list("lambda", "lambda"))
#  The log likelihood is computed using function dpois()
#  with the mle
      return(structure(list(estimate = estimate,
            sd = sds,
            vcov = vc, n = n,
            loglik = sum(dpois(x, estimate,log = TRUE))),                class = "f
    }
```

4(b) if-block

if (distname == "normal")

```
    if (distname == "normal") {
      if (!is.null(start))
        stop(gettextf("supplying pars for the %s distribution is not supported",
                      "Normal"), domain = NA)
# MLEs for mu and sigma are mx and sd0 (exact formulas)
      sd0 <- sqrt((n - 1)/n) * sd(x)
      mx <- mean(x)
      estimate <- c(mx, sd0)
# The sds of the MLEs are computed using exact formulas
#   for the normal mle mx and the scaled chi-square sd0^2
```

```
      sds <- c(sd0/sqrt(n), sd0/sqrt(2 * n))
      names(estimate) <- names(sds) <- c("mean", "sd")
# The variance/cov matrix of the estimates is diagonal
# (This is known from 18.650)

      vc <- matrix(c(sds[1]^2, 0, 0, sds[2]^2), ncol = 2,
                   dimnames = list(names(sds), names(sds)))
# The log likelihood is computed using r function dnorm()
      return(structure(list(estimate = estimate,
              sd = sds,vcov = vc, n = n,
              loglik = sum(dnorm(x, mx, sd0, log = TRUE))),
              class = "fitdistr"))
    }
```

4(c) if-block

if (distname == "gamma")

```
    if (distname == "gamma" && is.null(start)) {
      if (any(x < 0))
        stop("gamma values must be >= 0")
# If no starting values given, use method-of-moments
# estimates of the shape and scale parameters:
      m <- mean(x)
      v <- var(x)
      start <- list(shape = m^2/v, rate = m/v)
      start <- start[!is.element(names(start), dots)]
      control <- list(parscale = c(1, start$rate))
    }

    # The function fitdistr() uses these values with the optim(0
    #  function which uses the built-in dgamma function.
```

5. **ML estimation of Normal Linear Regression model using Newton's Method**

Consider

$$\vec{y} = X\vec{\beta} + \vec{\epsilon}, \;\; \vec{\epsilon} \sim N_n(\vec{0}, \sigma^2 \mathrm{I}_m)$$

where X is $n \times p$ design matrix and $\vec{\beta} \in R^p$ is regression parameter; $\sigma^2 > 0$ is error variance.

- Assume σ^2 known.
- Set $\vec{\beta}_0 \in R^p$ arbitrarily.

5(a) Give explicit solution for $\vec{\beta}_1$ using Newton algorithm.

5(b) Verify $\vec{\beta}_1$ equals $\hat{\beta}$, the MLE of β.

We have $\vec{y} \sim N_n(X\vec{\beta}, \sigma^2 \mathrm{I}_n)$ which has log likelihood

$$\ell(\beta) = -\tfrac{n}{2}[log(\sigma^2) + log(2\pi)] - \tfrac{1}{2\sigma^2}[\vec{y} - X\beta]^T[\vec{y} - X\beta]$$

For fixed σ^2, this is minimized by minimizing

$$\begin{aligned} g(\vec{\beta}) &= [\vec{y} - X\beta]^T[\vec{y} - X\beta] \\ &= \vec{y}^T\vec{y} - 2\vec{\beta}^T X^T[\vec{y}] + \beta^T X^T X\vec{\beta} \end{aligned}$$

For an initial guess $\vec{\beta}_0$, Newton's method solves for $\vec{\beta}_1$ in the equation:

$$\vec{0} = \nabla g(\vec{\beta}_1) \approx \nabla g(\vec{\beta}_0) + \nabla^2 g(\vec{\beta}_0)(\vec{\beta}_1 - \vec{\beta}_0)$$

where
$$\begin{aligned} \nabla g(\vec{\beta}) &= -2X^T(y - X\vec{\beta}) \\ \nabla^2 g(\vec{\beta}) &= +2X^T X \end{aligned}$$

This gives

$$\begin{aligned} \beta_1 &= \beta_0 - [\nabla^2 g(\vec{\beta}_0)]^{-1}\nabla[g(\vec{\beta}_0)] \\ &= \beta_0 - [2X^T X]^{-1}(-2X^T[\vec{y} - X\vec{\beta}_0]) \\ &= \beta_0 + [X^T X]^{-1}[X^T\vec{y} - X^T X\vec{\beta}_0] \\ &= \beta_0 + [X^T X]^{-1}X^T\vec{y} - \beta_0 \\ &= [X^T X]^{-1}X^T\vec{y} = \hat{\beta} \end{aligned}$$

This is the formula for the mle (see class).

5(c) Verify that solution independent of σ^2, so it applies for unknown σ^2.

Solution: Note that the likelihiood is maximized over $\vec{\beta}$ by $\hat{\beta}$ for any σ^2. Therefore, it is also the mle when σ^2 is unknown.