Statistical Computing 1, Stat 590 Homework 4 Name: Answer Key
Fall 2015 Prof. Erik B. Erhardt

Part I. (65 points) Do all calculations in LaTeX + R + knitr. Insert computer text output and graphics to support what you are saying. For this assignment, all R code should well commented and be visible (`echo=TRUE`) in the document where you have written it.

Goal: Construct a parametric bootstrap confidence interval for the coefficient of variation of waiting time to next eruption for the Old Faithful geyser using the rejection sampling method.

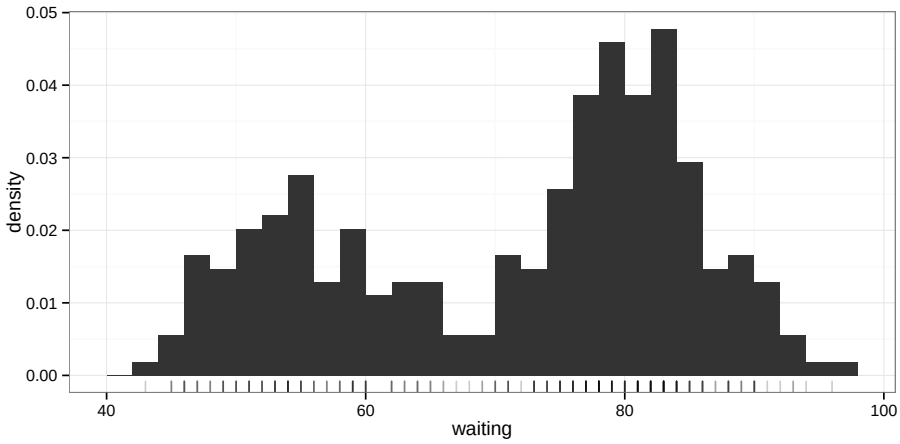
(15pts) **1.** The Old Faithful geyser dataset is in the `datasets` package.

```
library(datasets)
# ?faithful
# Old Faithful Geyser Data
# Waiting time between eruptions and the duration of the eruption for the
# Old Faithful geyser in Yellowstone National Park, Wyoming, USA.
#
# A data frame with 272 observations on 2 variables.
# [,1]   eruptions   numeric   Eruption time in mins
# [,2]   waiting     numeric   Waiting time to next eruption (in mins)

str(faithful)

## 'data.frame': 272 obs. of  2 variables:
##  $ eruptions: num  3.6 1.8 3.33 2.28 4.53 ...
##  $ waiting  : num  79 54 74 62 85 55 88 85 51 85 ...

head(faithful)

##   eruptions waiting
## 1     3.600      79
## 2     1.800      54
## 3     3.333      74
## 4     2.283      62
## 5     4.533      85
## 6     2.883      55

summary(faithful)

##    eruptions        waiting
##  Min.   :1.600   Min.   :43.0
##  1st Qu.:2.163   1st Qu.:58.0
##  Median :4.000   Median :76.0
##  Mean   :3.488   Mean   :70.9
##  3rd Qu.:4.454   3rd Qu.:82.0
##  Max.   :5.100   Max.   :96.0
```

(a) (5 pts) Plot the waiting time data and describe the pattern you see.

Solution: The data are clearly bimodal, and appear to be a mixture of two normal distributions.

```
library(ggplot2)
# Histogram overlaid with kernel density curve
p <- ggplot(faithful, aes(x = waiting))
  # Histogram with density instead of count on y-axis
p <- p + geom_histogram(aes(y=..density..), binwidth=2)
p <- p + geom_rug(alpha = 1/5)
p <- p + theme_bw()
print(p)
```



(b) (10 pts) A mixture distribution is of the form

$$f(x) \quad = \quad \sum_{i=1}^{k} \lambda_i f_i(x),$$

where λ_i is a proportional contribution of pdf $f_i(x)$ to the mixture $f(x)$. Look at the help for the `normalmixEM()` function in the `mixtools` package.

```
library(mixtools)
# ?normalmixEM

# generate some fake data to test the function
df.a <- data.frame(x = rnorm(100, mean = 2, sd = 1), dist = "A")
df.b <- data.frame(x = rnorm(200, mean = 9, sd = 2), dist = "B")
df.mix <- rbind(df.a, df.b)
df.mix$dist <- factor(df.mix$dist)

# inspect
summary(df.mix)
```

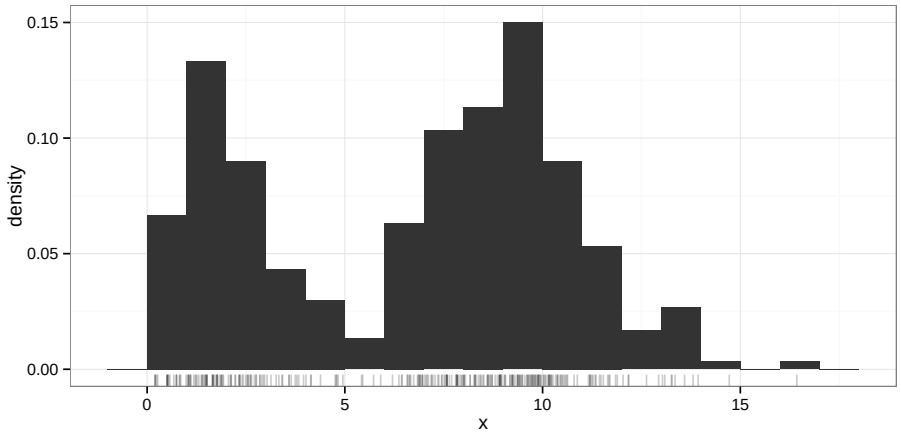
```
##        x           dist
##  Min.   : 0.2001   A:100
##  1st Qu.: 2.5599   B:200
##  Median : 7.5852
##  Mean   : 6.6410
##  3rd Qu.: 9.6965
##  Max.   :16.4273
```

```
# Inspect each distribution
```

```
library(plyr)
df.summary <- ddply(df.mix, 'dist', function(.subdf) {
    ## pull out the column of observations
    x <- .subdf$x
    data.frame( mean=mean(x), sd=sd(x), N=length(x))
})
df.summary
```

```
##   dist     mean        sd   N
## 1    A 1.852590 0.9875047 100
## 2    B 9.035222 2.1028740 200
```

```
# plot histogram of all data
library(ggplot2)
p <- ggplot(df.mix, aes(x = x))
p <- p + geom_histogram(aes(y=..density..), binwidth=1)
p <- p + geom_rug(alpha = 1/5)
p <- p + theme_bw()
print(p)
```



The parameters of the mixture distribution ($\lambda_i, \mu_i, \sigma_i$ for $i = 1, 2$) are well estimated with two normals when each component is based on a large sample size and the components are rather separate.

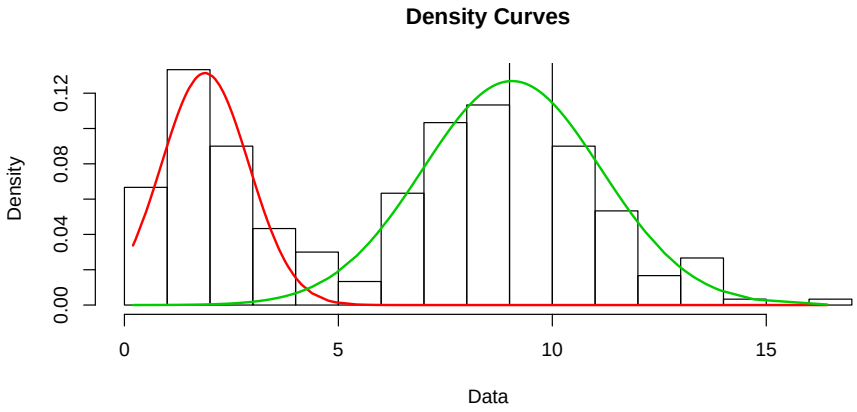
```
# estimate the mixture model parameters using the EM-algorithm
x.mix <- normalmixEM(df.mix$x)
```

```
## number of iterations= 67
```

```
x.mix[c("lambda", "mu", "sigma")]
```

```
## $lambda
## [1] 0.3369223 0.6630777
##
## $mu
## [1] 1.886775 9.056730
##
## $sigma
## [1] 1.022999 2.084844
```

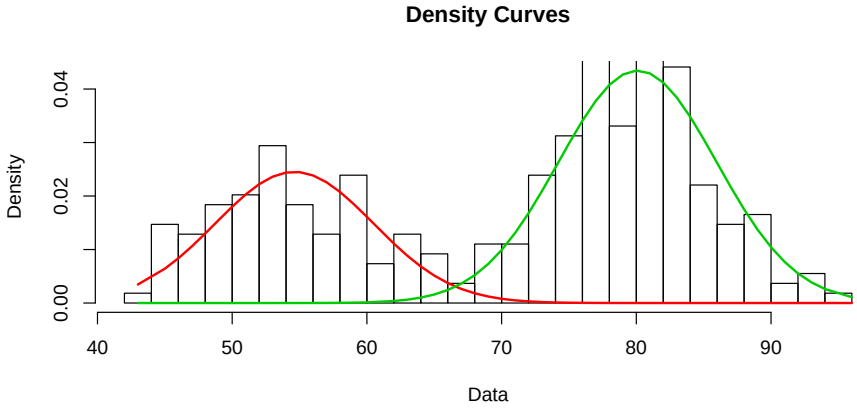
```
# ?plot.mixEM  # plotting options
plot(x.mix, which = 2, breaks = 20) # plot density components
```



Use this strategy to estimate the mixture distribution parameters for the waiting time. Interpret the parameters.

Solution:

```
library(mixtools)
wait.mix <- normalmixEM(faithful$waiting)
## number of iterations= 34
wait.mix[c("lambda", "mu", "sigma")]
## $lambda
## [1] 0.3608869 0.6391131
##
## $mu
## [1] 54.61489 80.09109
##
## $sigma
## [1] 5.871243 5.867717
plot(wait.mix, which=2, breaks = 20) # plot density components
```



Roughly a third of the waiting times are centered at 54.6 and two thirds at 80.1 both with standard deviations roughly 5.87. The mixture distribution is estimated to be

$$f(x|\hat{\lambda}_1, \hat{\mu}_1, \hat{\sigma}_1, \hat{\lambda}_2, \hat{\mu}_2, \hat{\sigma}_2) \quad = \quad \hat{\lambda}_1 \text{Normal}(x|\hat{\mu}_1, \hat{\sigma}_1^2) + \hat{\lambda}_2 \text{Normal}(x|\hat{\mu}_2, \hat{\sigma}_2^2).$$

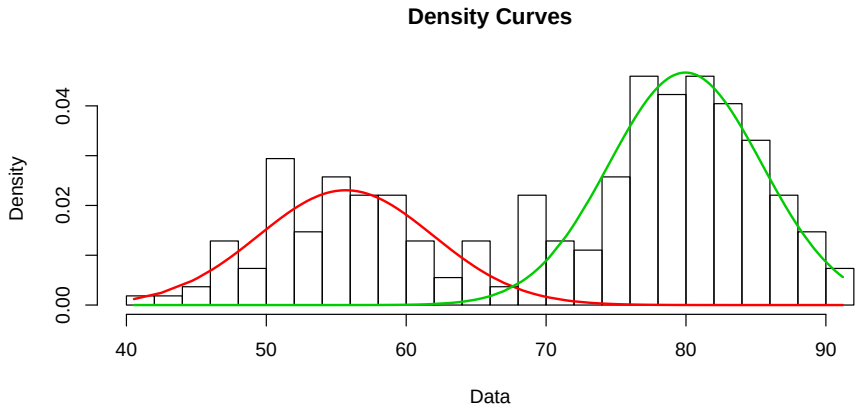
(30pts) **2. Simulating random deviates from a mixture distribution**

We will consider two strategies for drawing random deviates from the mixture distribution in the previous problem.

(a) (10 pts) Using the estimated parameters of the mixture distribution ($\hat{\lambda}_i, \hat{\mu}_i, \hat{\sigma}_i$ for $i = 1, 2$), one stategy is to draw a random deviate from each of the component distributions (Normal$(x|\hat{\mu}_1, \hat{\sigma}_1^2)$ or Normal$(x|\hat{\mu}_2, \hat{\sigma}_2^2)$) with probability proportional to their proportional contribution to the mixture ($\hat{\lambda}_1$ and $\hat{\lambda}_2$). Write code to simulate from the fitted mixture distribution using this strategy and plot a histogram based on a sample size equal to the original sample.

Solution: The plot below indicates the sample drawn using this strategy is similar to the original data.

```
# sample size
n <- nrow(faithful)
# draw n uniform random numbers to determine which Normal component to draw from
u <- runif(n)
# count how many to draw from Normal_1
n1 <- sum( u < wait.mix$lambda[1])
# the rest are drawn from Normal_2
n2 <- n - n1
c(n1, n2)
## [1]  97 175
# draw sample
y <- c( rnorm(n1, mean = wait.mix$mu[1], sd = wait.mix$sigma[1]),
        rnorm(n2, mean = wait.mix$mu[2], sd = wait.mix$sigma[2]))

library(mixtools)
y.mix <- normalmixEM(y)
## number of iterations= 44
y.mix[c("lambda", "mu", "sigma")]
## $lambda
## [1] 0.3600008 0.6399992
##
## $mu
## [1] 55.68335 79.95964
##
## $sigma
## [1] 6.230077 5.468943
plot(y.mix, which=2, breaks = 20) # plot density components
```



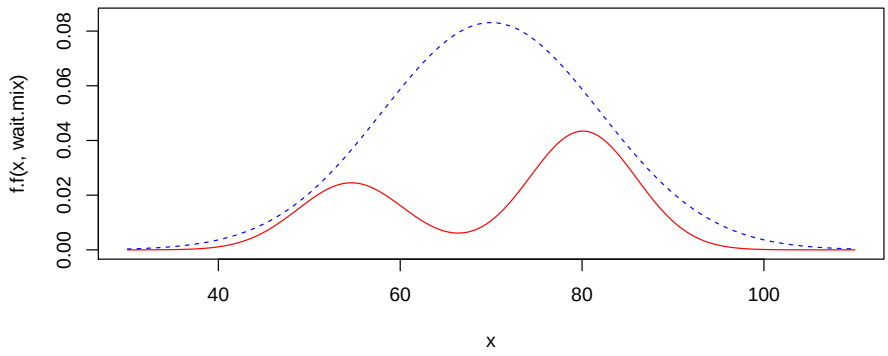
(b) (10 pts) The rejection sampling method can be used.
Set up the distributions needed for rejection sampling.

1. Using the estimated parameters of the mixture distribution ($\hat{\lambda}_i, \hat{\mu}_i, \hat{\sigma}_i$ for $i = 1, 2$) write a `function()` for the fitted density function $f(x)$ of the mixture distribution.
2. Determine a density which is easy to sample from, $h(x)$, and scale factor α to construct an envelope function $e(x) \equiv h(x)/\alpha$.
3. Show that this envelope function $e(x)$ is strictly not less than $f(x)$ over a sensible domain.

Solution:

```
## (1)
# function f(x) is a mixture of normals parametrized by the fitted mixture distribution
f.f <- function(x, param) {
  f <- param$lambda[1] * dnorm(x, mean = param$mu[1], sd = param$sigma[1]) +
       param$lambda[2] * dnorm(x, mean = param$mu[2], sd = param$sigma[2])
  return(f)
}

## (2)
# let h(x) be a normal distribution
f.h <- function(x, param) {
  h <- dnorm(x, mean = param$mu, sd = param$sigma)
  return(h)
}
# parameters of h(x)
h.param <- data.frame(mu = 70, sigma = 12)


# let's try many alphas
alphas <- seq(from=0.1, to=1, by=0.1)

# for each alpha, find number of times f(x) exceeds envelope
alpha.test <- ldply( alphas, function(.alpha) {
    envelope <- f.h(x, h.param)/.alpha
```

```
    density <- f.f(x, wait.mix)
    data.frame( alpha=.alpha, failures=sum( density>envelope))
})

# show results
alpha.test
##    alpha failures
## 1    0.1        0
## 2    0.2        0
## 3    0.3        0
## 4    0.4        0
## 5    0.5   845007
## 6    0.6  1835923
## 7    0.7  2385607
## 8    0.8  2789413
## 9    0.9  3110510
## 10   1.0  3377605
# determine alpha scale parameter
alpha <- 0.4

## (3)
# plot density function f(x)
x <- seq(30, 110, length=200)
plot(x, f.f(x, wait.mix), type = "l", col = "red", ylim = c(0,0.085))
# plot envelope function e(x) = h(x)/alpha
points(x, f.h(x, h.param) / alpha, type = "l", col = "blue", lty = 2)
```



By visual inspection, over the meaningful range of these distributions, the (dashed blue) envelope function, $e(x)$, is strictly larger than the (solid red) distribution of interest, $f(x)$.

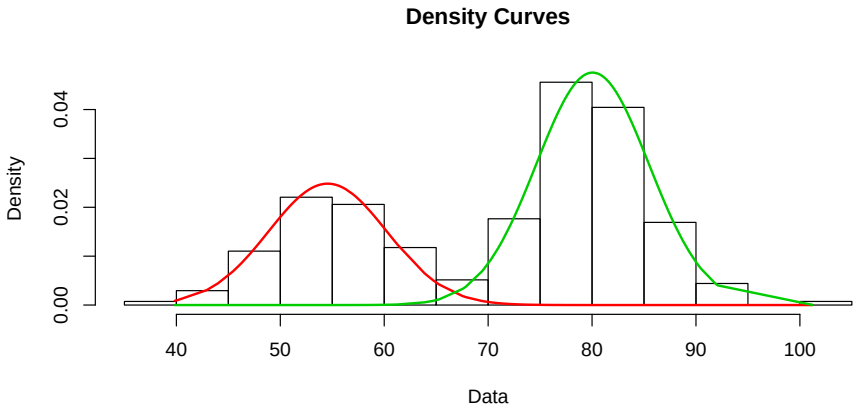
(c) (10 pts) Rejection sampling method, continued...
Perform the rejection sampling.

1. Draw x from the proposal distribution, $h(x)$.
2. Draw u from a uniform distribution.
3. Determine, using the rejection rule, whether to reject or accept x.
4. Repeat this until you have $n = 272$ accepted samples.

5. Plot a histogram of the samples.

Solution: The rejection method plot below indicates the sample drawn using this strategy is similar to the original data.

```
# initialize a sample vector to hold accepted samples
sam <- rep(NA, n)

# continue to perform rejection sampling until we accept n samples
n.accepted <- 0
while (n.accepted < n) {
  # sample from h(x)
  x <- rnorm(1, mean = h.param$mu, sd = h.param$sigma)
  # sample from uniform
  u <- runif(1)
  # evaluate envelope distribution
  e <- f.h(x, h.param) / alpha
  # evaluate target distribution
  f <- f.f(x, wait.mix)
  # determine whether we reject this sample
  reject <- ( u > f / e )
  # if we didn't reject this sample
  if (!reject) {
    # increment counter
    n.accepted <- n.accepted + 1
    # put sampled x into sample
    sam[n.accepted] <- x
  }
}

# plot a histogram of the samples
library(mixtools)
sam.mix <- normalmixEM(sam)
## number of iterations= 23
sam.mix[c("lambda", "mu", "sigma")]
## $lambda
## [1] 0.3542179 0.6457821
##
## $mu
## [1] 54.55998 80.05367
##
## $sigma
## [1] 5.690489 5.416939
plot(sam.mix, which=2, breaks = 20) # plot density components
```



(20pts) **3. Parametric bootstrap**

(a) (10 pts) Using your results from #2, write a function to sample random deviates using the rejection sampling method. This function should have the following arguments: N, the number of samples; f.h, the proposal distribution function (and all associated parameters); alpha; and f.f, the target distribution evaluation function (and all associated parameters). It should return a vector of N deviates from the target distribution. A vectorized version of the above function is preferred. Half of the points will be based coding on style – write an efficient, organized, and well-documented function for full points.

Solution:

```
# Since we are drawing many samples, a faster way will be better
# The vectorized version of sampling below is *much* faster than the while() used earlier!

# number of samples
R <- 1e4

# n, from before, is the size of each sample

# number of samples to draw (10% more than we expect to require)
# scaled by alpha, the relative integral area of f(x) and e(x)
RR <- round(1.1 * (n * R / alpha))

# sample from h(x)
x <- rnorm(RR, mean = h.param$mu, sd = h.param$sigma)
# sample from uniform
u <- runif(RR)
# evaluate envelope distribution
e <- f.h(x, h.param) / alpha
# evaluate target distribution
f <- f.f(x, wait.mix)
# determine whether we reject this sample
reject <- ( u > f / e )
# if we didn't reject this sample, assign those values to accepted
```

```
sam.accepted <- x[!reject]
# this should be at least R
if (length(sam.accepted) < n*R) {
  cat("NOTE: Not enough samples, make RR larger.")
}
# our sample are the first n*R accepted values of those accepted
sam <- sam.accepted[1:(n*R)]
```

(b) (10 pts) Use the above function to perform a parametric bootstrap to calculate $R = 10^4$ bootstrap values of the coefficient of variation and compute a central 95% CI.

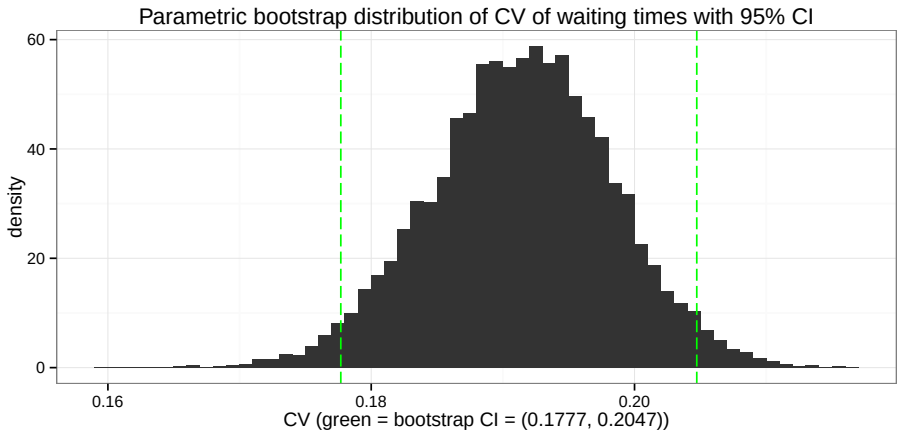
Solution:

```
# reshape samples to have n columns
sam2 <- matrix(sam, ncol = n)

# create function to calculate the coefficient of variation
f.cv <- function(x) {
  cv <- sd(x)/mean(x)
  return(cv)
}

# calculate the coefficient of variation of each row of sam2
cv.sam2 <- apply(sam2, MARGIN = 1, f.cv)

# 0.025th and 0.975th quantile gives equal-tail bootstrap CI
CI.bs <- quantile(cv.sam2, probs = c(0.025, 0.975))
CI.bs
```

```
##      2.5%     97.5%
## 0.1776896 0.2047279
```

```
## Plot the bootstrap distribution with CI
# First put data in data.frame for ggplot()
dat.cv <- data.frame(cv.sam2)

library(ggplot2)
p <- ggplot(dat.cv , aes(x = cv.sam2))
p <- p + geom_histogram(aes(y=..density..), binwidth=0.001)
# vertical lines at CI
p <- p + geom_vline(aes(xintercept=CI.bs[1]), colour="green", linetype="longdash")
p <- p + geom_vline(aes(xintercept=CI.bs[2]), colour="green", linetype="longdash")
p <- p + labs(title = "Parametric bootstrap distribution of CV of waiting times with 95% CI")
p <- p + xlab(paste("CV (green = bootstrap CI = (", signif(CI.bs[1], 4), ", ",
                              signif(CI.bs[2], 4), "))", sep=""))
p <- p + theme_bw()
print(p)
```

```
## Warning in data.frame(xintercept = structure(0.17768955801234, .Names = "2.5%"),
:  row names were found from a short variable and have been discarded
## Warning in data.frame(xintercept = structure(0.204727891023878, .Names = "97.5%"),
:  row names were found from a short variable and have been discarded
```



The 95% parametric bootstrap CI using the rejection method of sampling from the mixture distribution gives CI $= (0.1777, 0.2047)$.