Lab 4

Hanbo Li

January 26, 2016

This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see http://rmarkdown.rstudio.com.

When you click the **Knit** button a document will be generated that includes both content as well as the output of any embedded R code chunks within the document. You can embed an R code chunk like this:

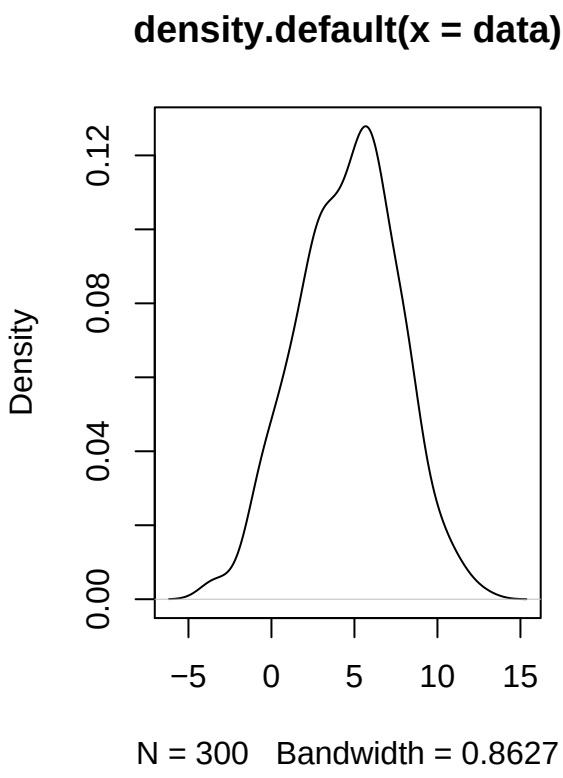
sampling in R:

sample(x, size, replace, prob)

```
##  [1]  6  9  5  3 10  7  2  1  4  8
```

```
##  [1]  2  1  2  3  9  1  1 10  6  6
```

```
##  [1] 4.8673668 3.9080140 2.4477536 0.4813591 8.3525866 1.0640444 0.8713677
##  [8] 6.5746737 6.0720086 0.9817580
```



bootstrap

```
install.packages("Lock5Data",repos="http://cran.us.r-project.org")
```

```
##
## The downloaded binary packages are in
##  /var/folders/83/c7nm5x8d7pb6ggq68npmggwr0000gn/T//Rtmp63Zqhv/downloaded_packages
```

```r
library(Lock5Data)
data(CommuteAtlanta)
str(CommuteAtlanta)
```

```
## 'data.frame':    500 obs. of  5 variables:
##  $ City    : Factor w/ 1 level "Atlanta": 1 1 1 1 1 1 1 1 1 1 ...
##  $ Age     : int  19 55 48 45 48 43 48 41 47 39 ...
##  $ Distance: int  10 45 12 4 15 33 15 4 25 1 ...
##  $ Time    : int  15 60 45 10 30 60 45 10 25 15 ...
##  $ Sex     : Factor w/ 2 levels "F","M": 2 2 2 1 1 2 2 1 2 1 ...
```

Task: Construct the confidence interval for the mean commute time in Atlanta.

step 1: Estimate mean.

```r
time.mean = mean(CommuteAtlanta$Time)
time.mean
```

```
## [1] 29.11
```

step 2: To find the standard error, we will use bootstrap method. We store each bootstrap sample as a row of a matrix. For example, in this case, the number of data in CommuteAtlanta$Time is 500. For each time, we sample with replacement 500 data to form a bootstrap sample, and we repeat this for 1000 times, i.e. we get totally 1000 bootstrap samples.

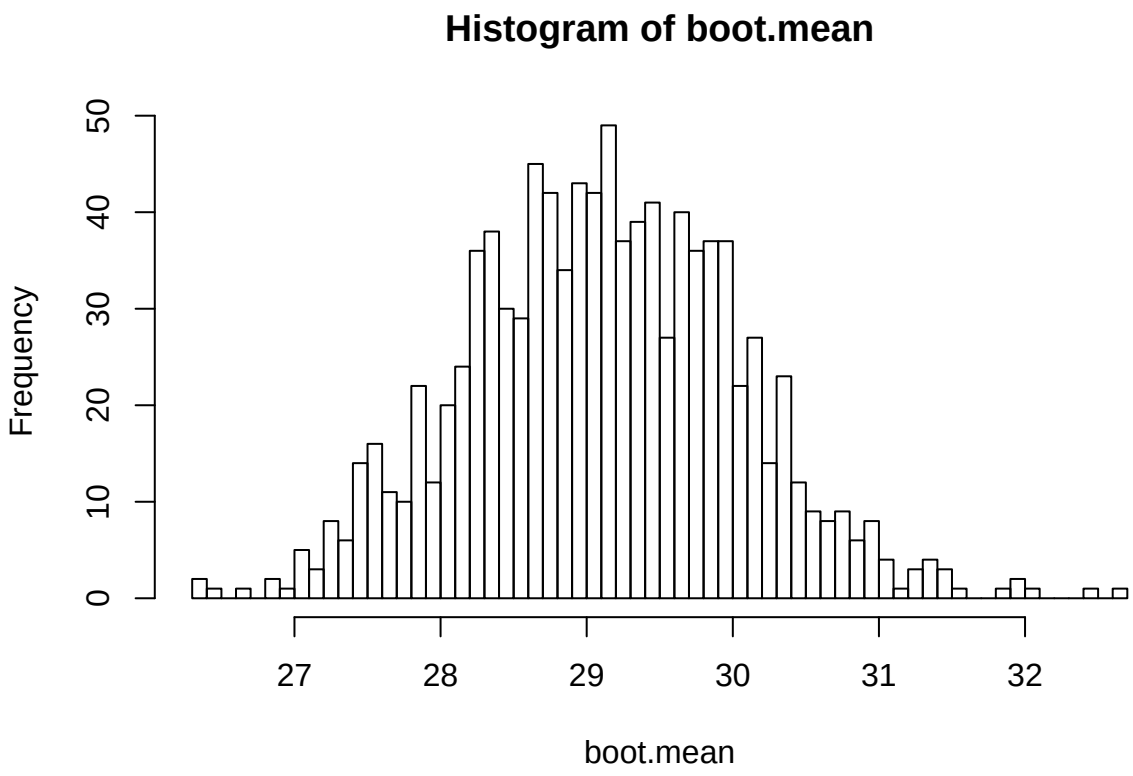
```r
B = 1000 # the number of bootstrap samples we want
n = nrow(CommuteAtlanta) # how many data we have, or we can use
n = length(CommuteAtlanta$Time)
boot.samples = matrix(sample(CommuteAtlanta$Time, size = B * n, replace = TRUE), B, n)
```

Then for each bootstrap sample (i.e. each row of the matrix above), we find its mean by using apply() function in R. This function is very useful.

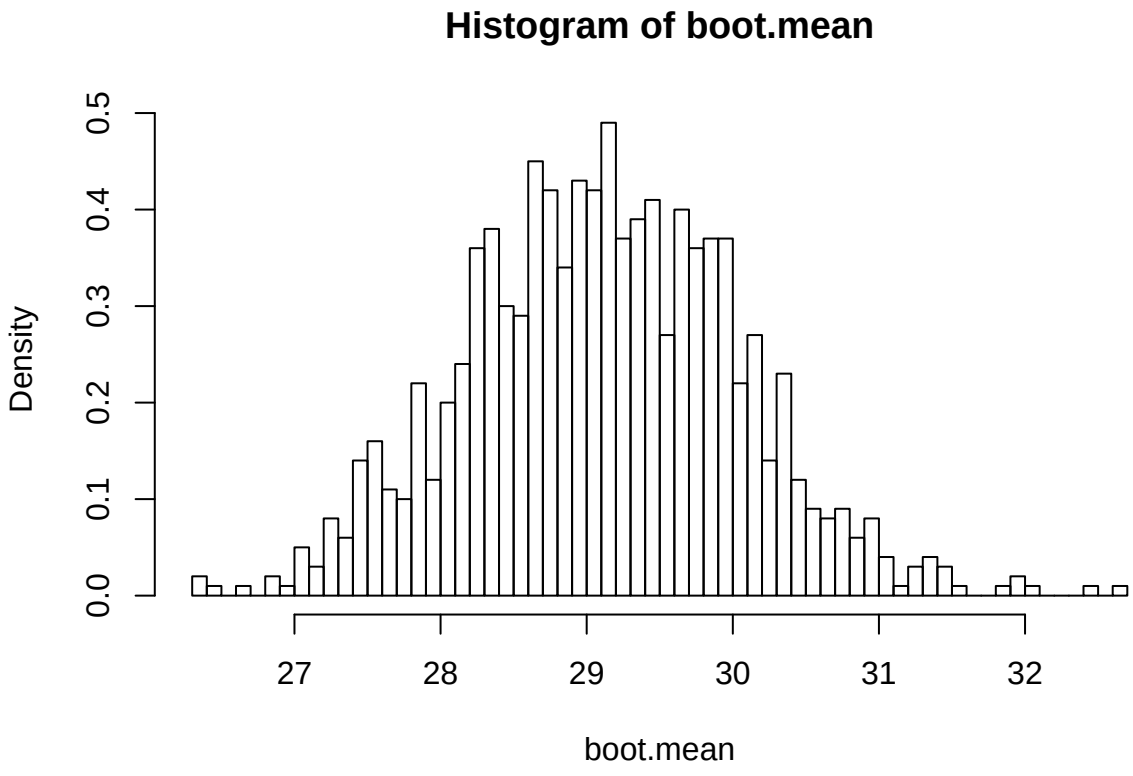
```r
boot.mean = apply(boot.samples, 1, mean) # 1 means applying to each row. 2 means applying to each colum
head(boot.mean)
```

```
## [1] 29.292 29.764 29.572 28.368 30.180 28.936
```

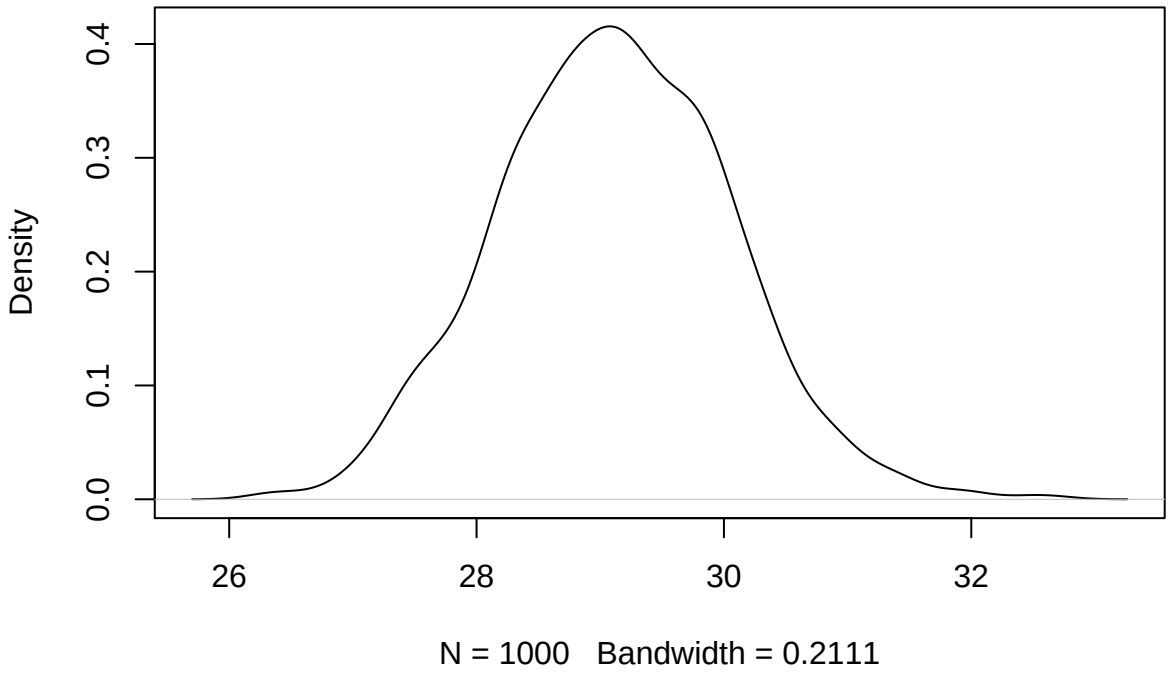
```r
hist(boot.mean, breaks = 50, freq=TRUE) # show frequency
```




```
hist(boot.mean, breaks = 50, freq=FALSE) # show density
```




```
plot(density(boot.mean), main="density of bootstrap sample means")
```



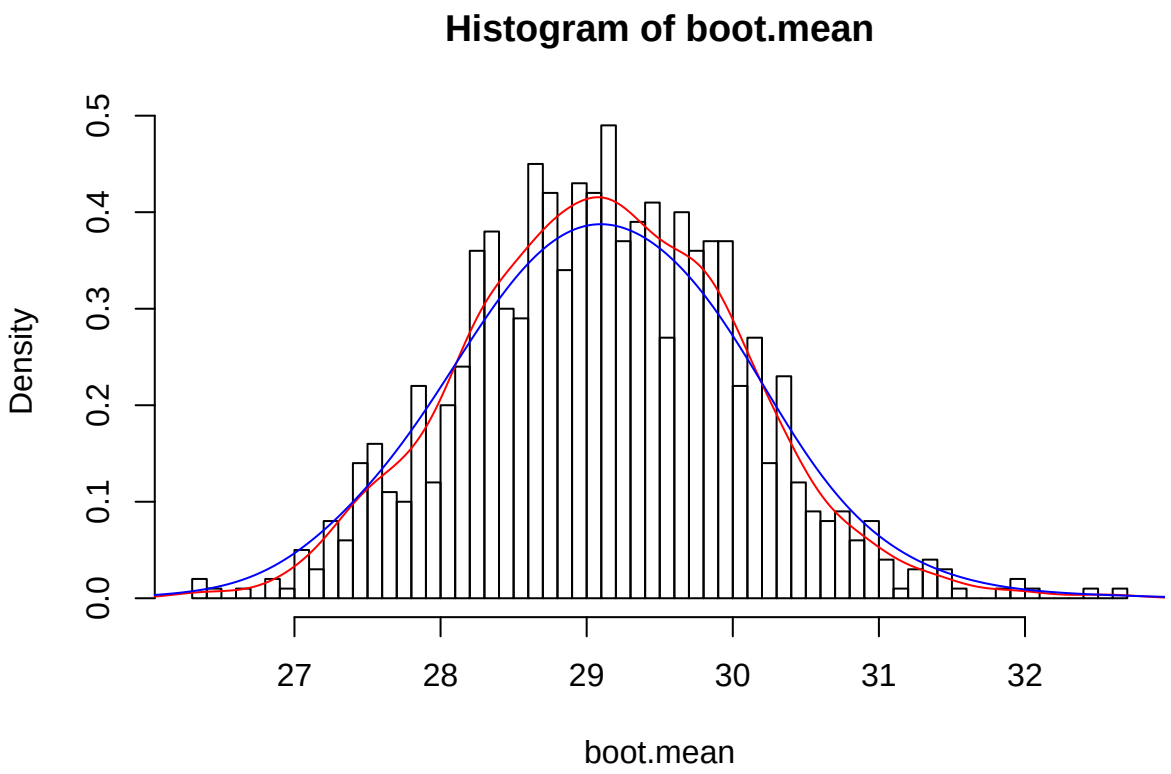

```
# put these two plots together. Don't use plot(), use lines() instead.
hist(boot.mean, breaks = 50, freq=FALSE)
lines(density(boot.mean), col="red")
lines(density(boot.mean, adjust=2), col="blue") # smooth the density

# another way to do it: ggplot!
install.packages("ggplot2", repos="http://cran.us.r-project.org")
```

```
##
## The downloaded binary packages are in
##  /var/folders/83/c7nm5x8d7pb6ggq68npmggwr0000gn/T//Rtmp63Zqhv/downloaded_packages
```

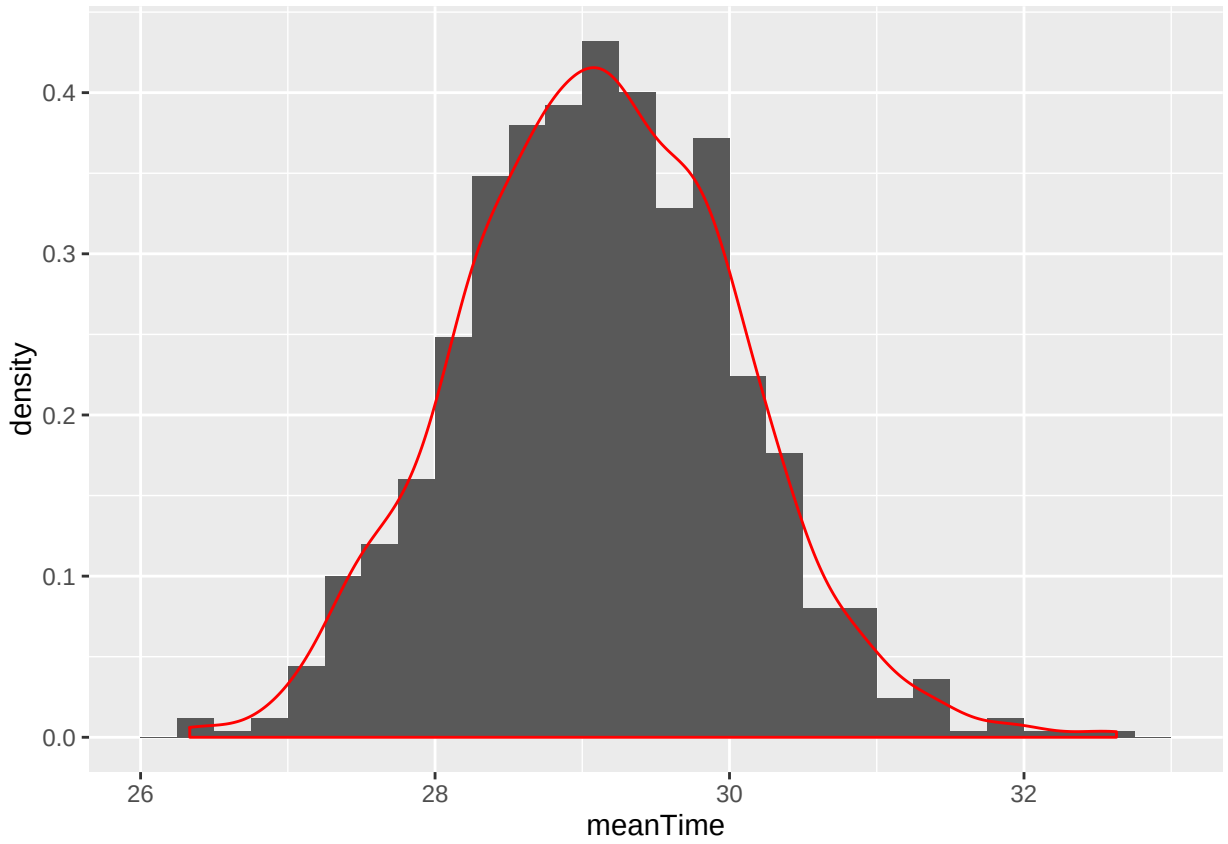
```
require(ggplot2)
```

```
## Loading required package: ggplot2
```

```
## Warning: package 'ggplot2' was built under R version 3.1.3
```




```
ggplot(data.frame(meanTime = boot.mean),aes(x=meanTime)) +
geom_histogram(binwidth=0.25,aes(y=..density..)) +
geom_density(color="red")
```



Finally, standard error for the bootstrap means!

```
time.se = sd(boot.mean)
time.se
```

```
## [1] 0.9337494
```

step 3: Now we have mean, we have the se of means, we can construct the confidence interval.

```
me = 1.96*time.se
time.mean + c(-1, 1) * me
```

```
## [1] 27.27985 30.94015
```

Therefore, we are 95% confident that the mean commute time in Atlanta among commuters who do not work at home is in the interval from 27.3 to 30.9 minutes.

The accuracy of this inference depends on the original sample being representative from the population of interest.

Now, let look at the data for HW 2.

```
videodata <- read.table("videodata.txt", header=TRUE) # read the first line as header
head(videodata)
```

```
##   time like where freq busy educ sex age home math work own cdrom email
```

```
## 1  2.0    3     3    2    0    1   0  19    1    0   10   1     0     1
## 2  0.0    3     3    3    0    0   0  18    1    1    0   1     1     1
## 3  0.0    3     1    3    0    0   1  19    1    0    0   1     0     1
## 4  0.5    3     3    3    0    1   0  19    1    0    0   1     0     1
## 5  0.0    3     3    4    0    1   0  19    1    1    0   0     0     1
## 6  0.0    3     2    4    0    0   1  19    0    0   12   0     0     0
##   grade
## 1     4
## 2     2
## 3     3
## 4     3
## 5     3
## 6     3
```

```
names(videodata)
```

```
##  [1] "time"  "like"  "where" "freq"  "busy"  "educ"  "sex"   "age"
##  [9] "home"  "math"  "work"  "own"   "cdrom" "email" "grade"
```

```
dim(videodata)
```

```
## [1] 91 15
```

```
n = dim(videodata)[1]
```

Let's look at the proportion of females in the data.

```
male_num <- sum(videodata$sex)
female_num <- sum(!videodata$sex)

female_proportion <- female_num/n
female_proportion
```

```
## [1] 0.4175824
```

By lecture notes, we know the estimate of standard error of female proportion is se = 0.044. (See page 36 and page 43.)

```
x_bar <- female_proportion
n = nrow(videodata)
N = 314
se_estimator <- sqrt(x_bar*(1-x_bar)/(n-1))*sqrt((N-n)/N)
se_estimator
```

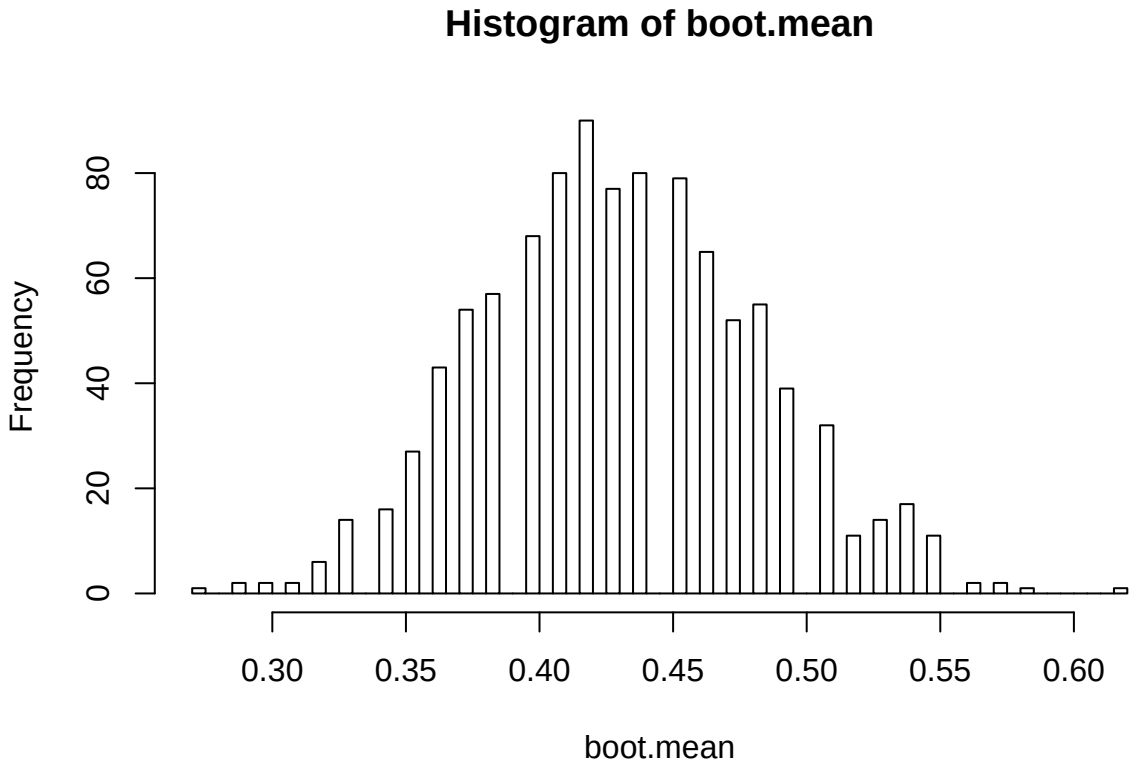
```
## [1] 0.04380813
```

Let us use bootstrap to verify. (Bootstrap for survey: First create a bootstrap population, then draw w/o replacement from this population.)

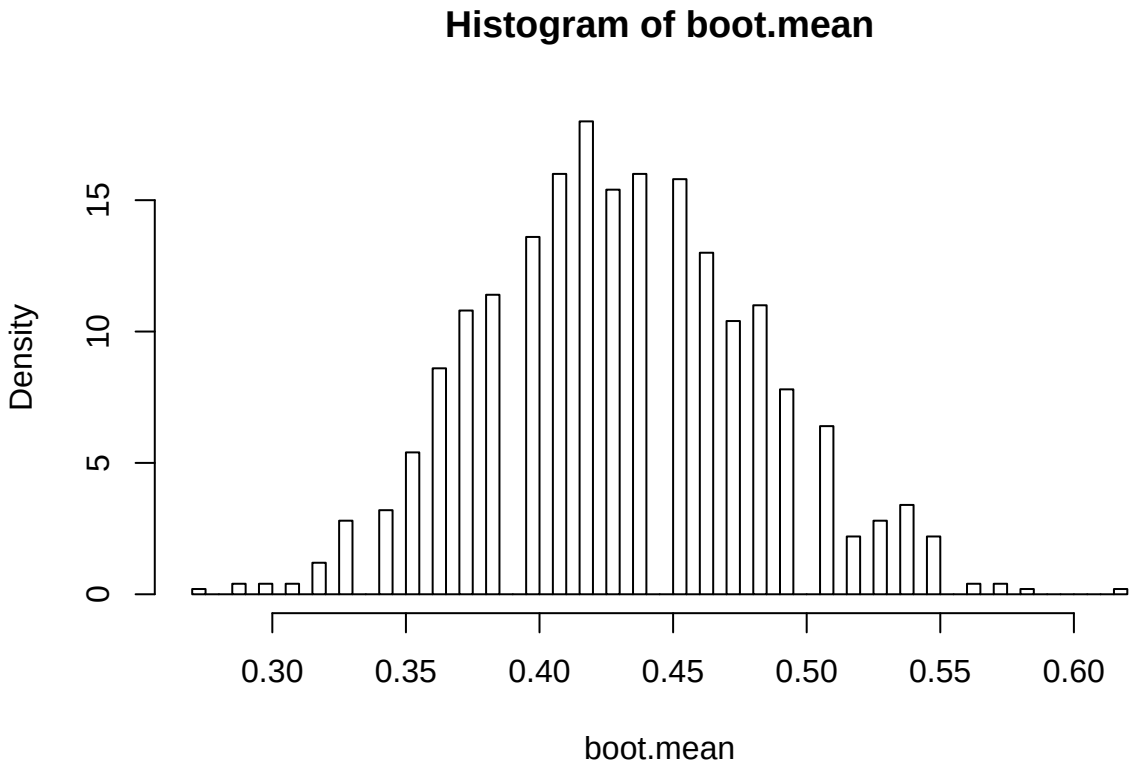
```r
N = 314
B = 1000
n = nrow(videodata)

# bootstrap population
boot.population = sample(videodata$sex, size=N, replace=TRUE)

# draw from bootstrap population
boot.samples = matrix(sample(boot.population, size = B * n, replace = TRUE), B, n)

# Now 1 means male and 0 means female. For convenience, let's change 1 to mean female.
boot.samples = !boot.samples
boot.mean = apply(boot.samples, 1, mean)
head(boot.mean)
```

```
## [1] 0.4285714 0.4835165 0.4175824 0.4175824 0.4835165 0.3846154
```

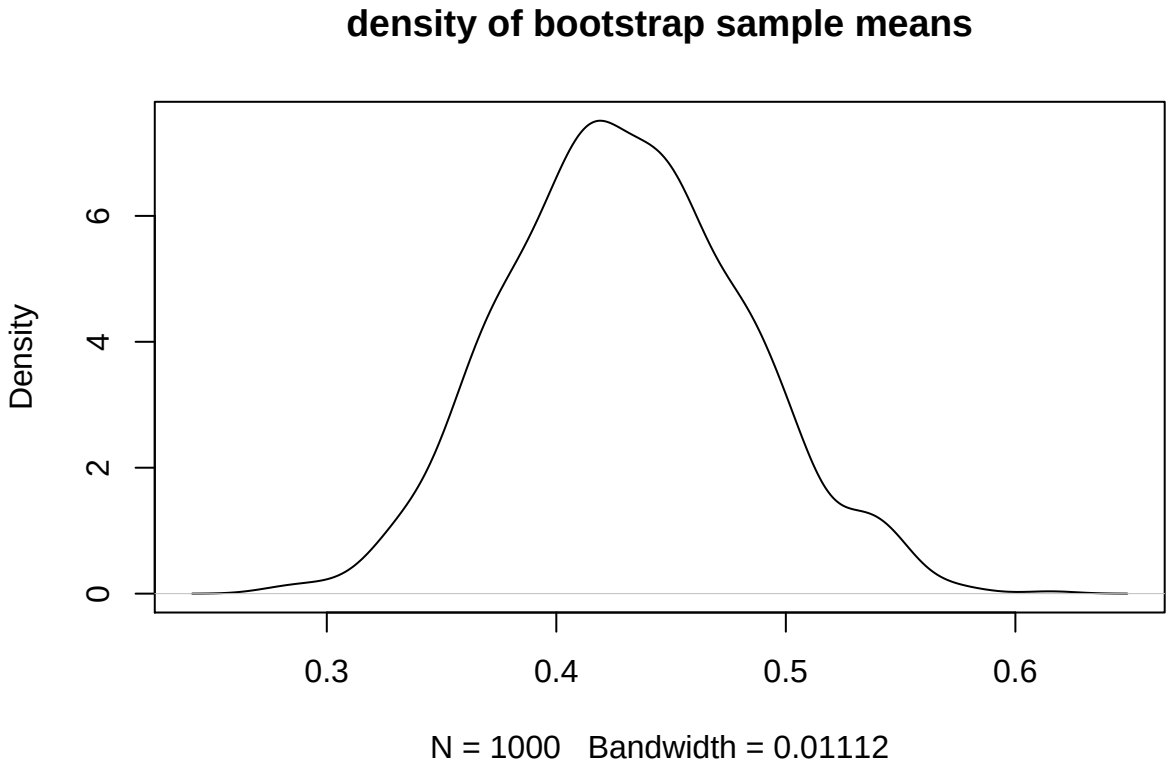
```r
hist(boot.mean, breaks = 50, freq=TRUE) # show frequency
```



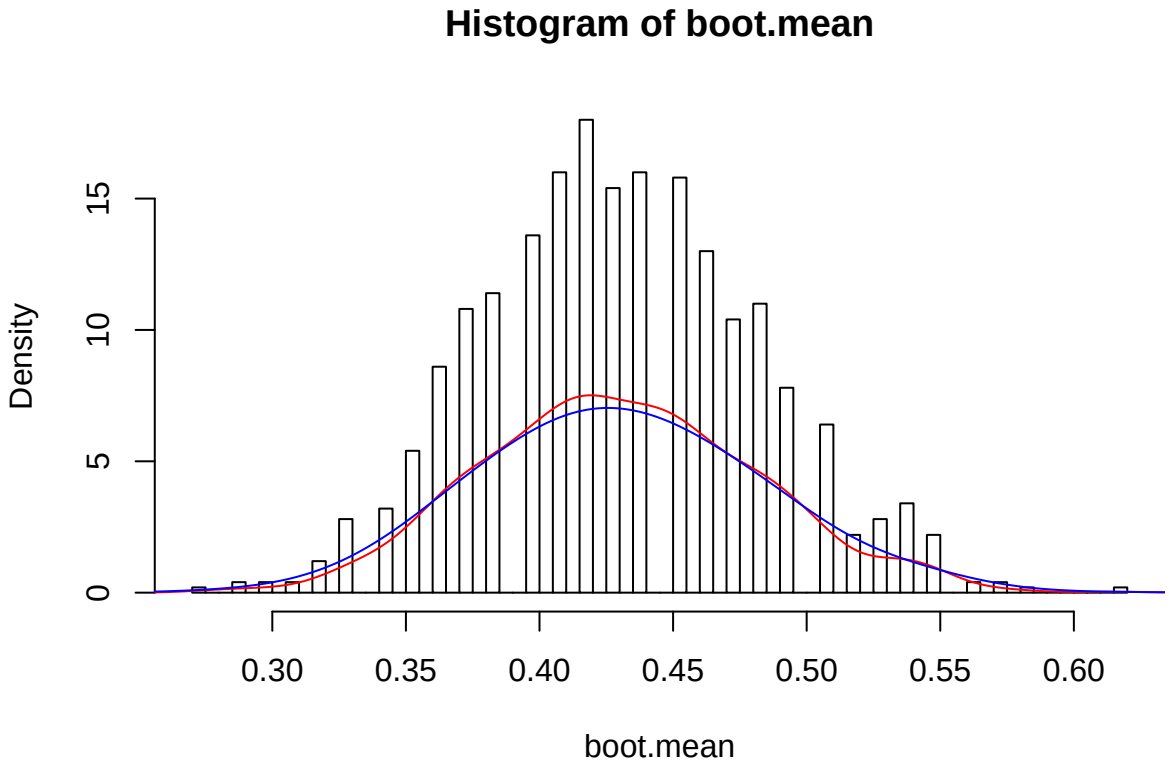

```r
hist(boot.mean, breaks = 50, freq=FALSE) # show density
```




```
plot(density(boot.mean), main="density of bootstrap sample means")
```



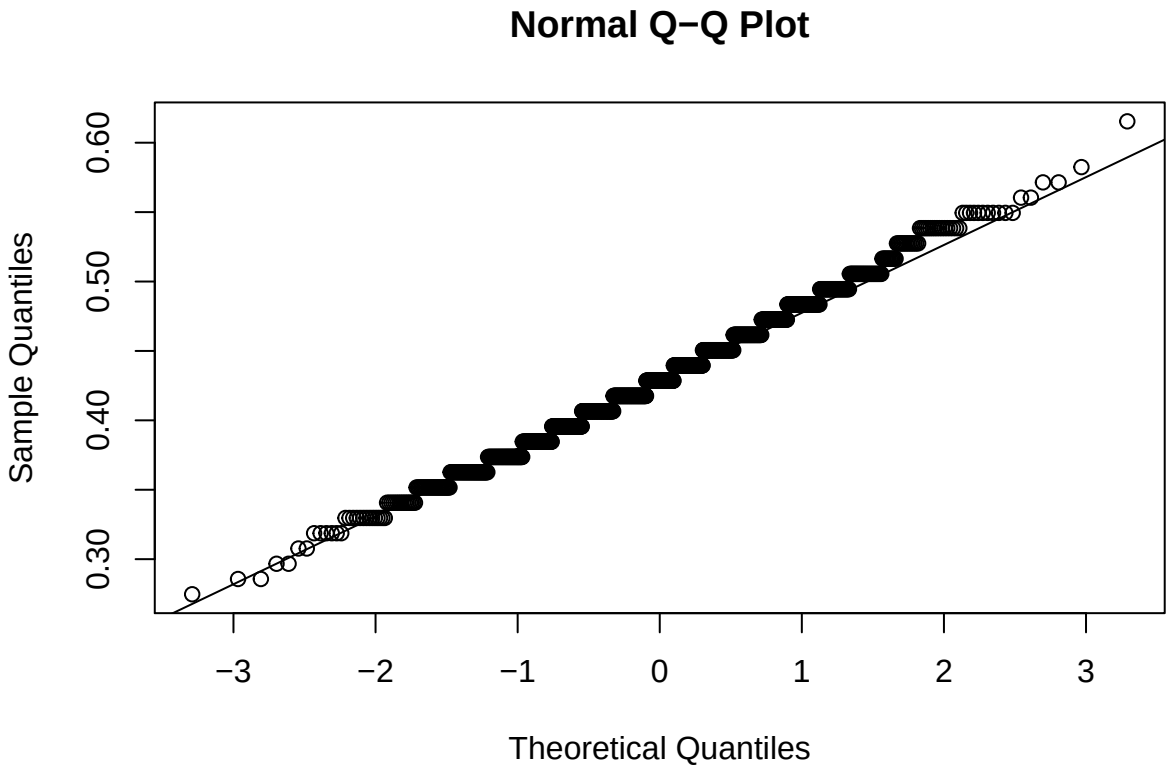
```r
# put these two plots together. Don't use plot(), use lines() instead.
hist(boot.mean, breaks = 50, freq=FALSE)
lines(density(boot.mean), col="red")
lines(density(boot.mean, adjust=2), col="blue") # smooth the density
```




```r
female.se = sd(boot.mean)
female.se
```

```
## [1] 0.05167546
```

```r
qqnorm(boot.mean)
qqline(boot.mean)
```

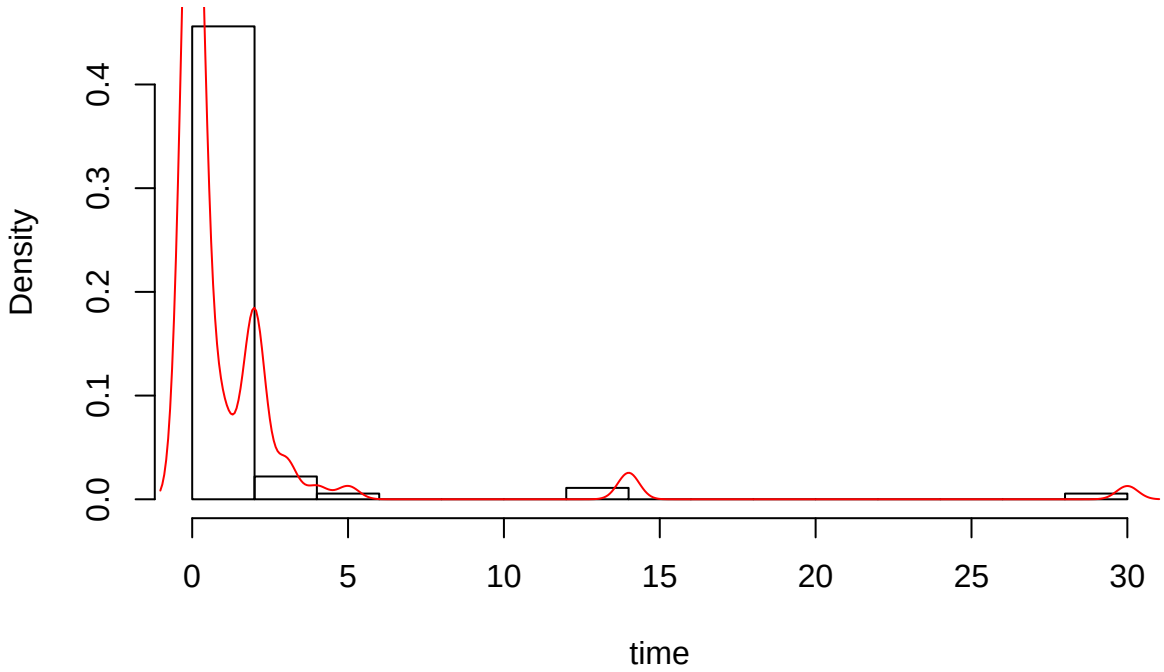


se $=$ 0.051

Note that: two methods are equivalent. Drawing 314 bootstrap samples w/ replacement from 91 data, then draw 91 samples w/o replacement from these 314 samples is equivalent to (under expectation) drawing 91 bootstrap samples w/ replacement from 91 data.

Time spent on videogames

```
time <- videodata$time
hist(time, breaks=20, freq=FALSE)
lines(density(time), col="red")
```



(Lecture note, page 45) From the histogram of the time spent playing video games by the students in the sample, we see that the sample distribution is extremely skewed.

This observation raises a question of whether the probability distribution of the sample average follows normal curve.

```
time_avg <- mean(time)
time_avg
```

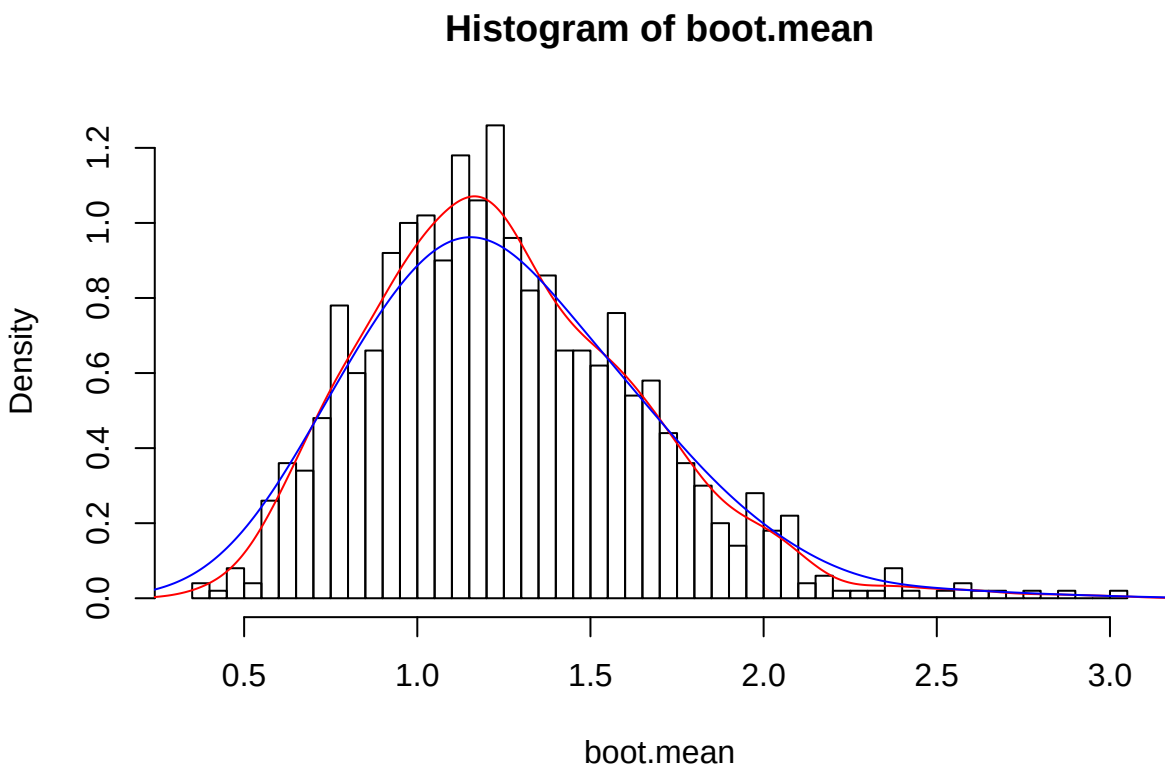
```
## [1] 1.242857
```

Again, bootstrap!

```
B = 1000
n = nrow(videodata)
boot.samples = matrix(sample(time, size = B * n, replace = TRUE), B, n)
boot.mean = apply(boot.samples, 1, mean)
head(boot.mean)
```

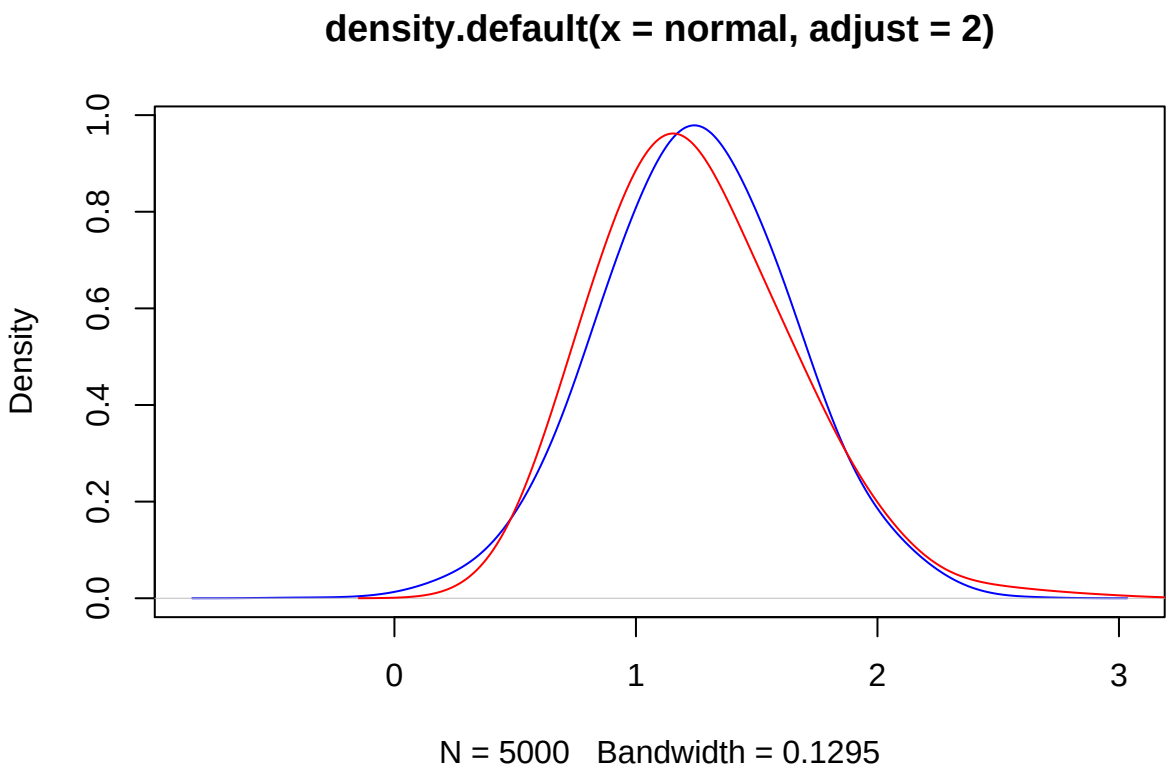
```
## [1] 1.2109890 1.0835165 1.5780220 0.3868132 1.6538462 2.0450549
```

```
hist(boot.mean, breaks = 50, freq=FALSE)
lines(density(boot.mean), col="red")
lines(density(boot.mean, adjust=2), col="blue") # smooth the density
```



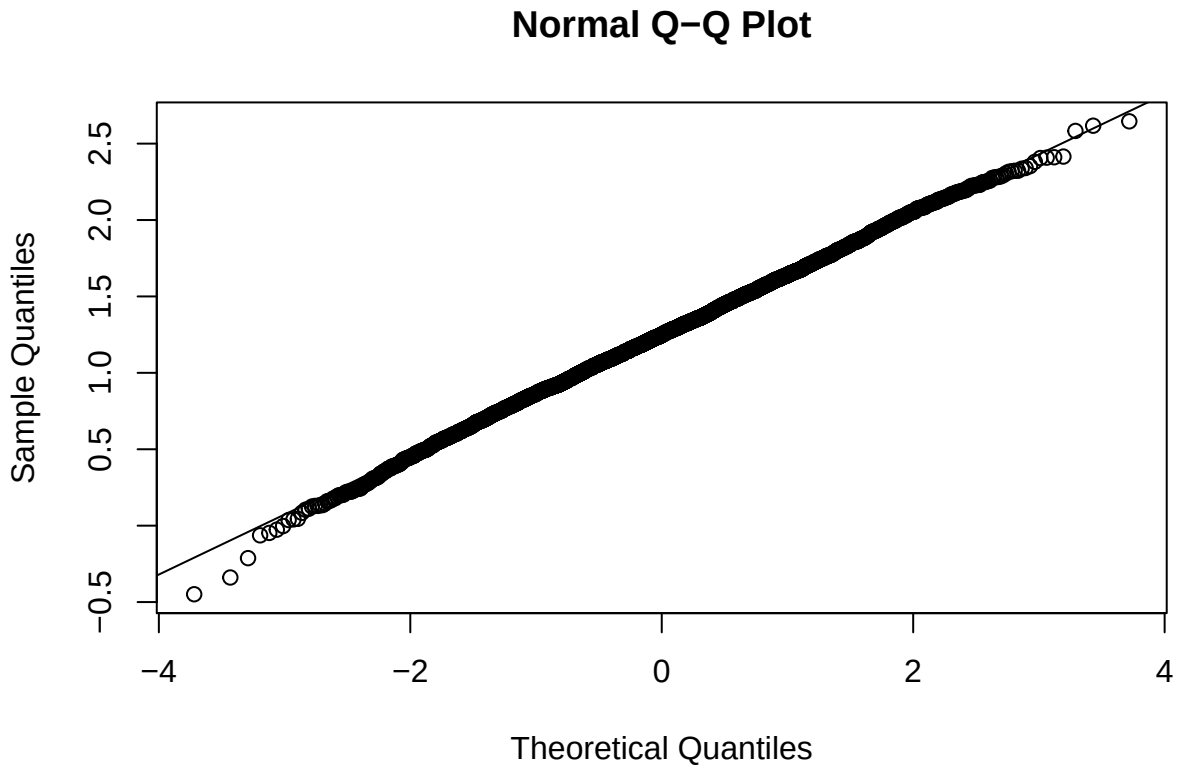
Not that normal. Let's compare the density with normal density with the same mean and sd.

```
mu <- mean(boot.mean)
sigma <- sd(boot.mean)
normal <- rnorm(5000,mu,sigma)
plot(density(normal, adjust=2), col="blue")
lines(density(boot.mean, adjust=2), col="red")
```

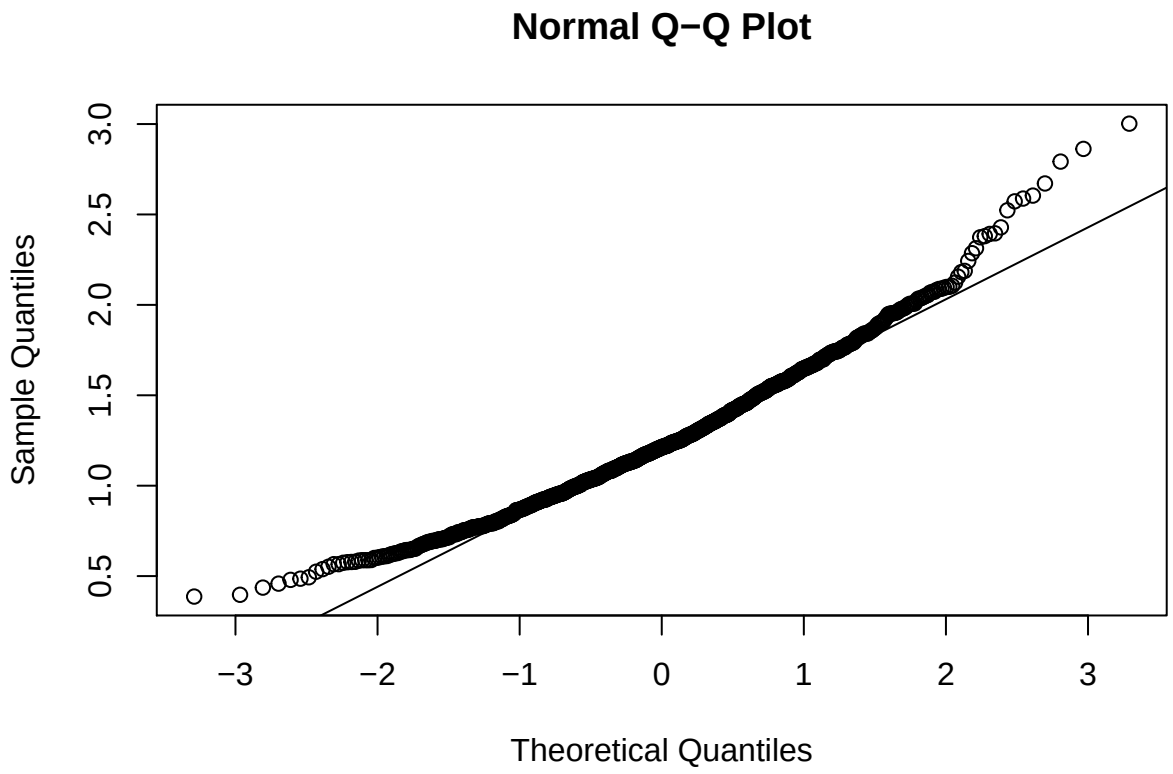


Q-Q plot

```
# What does the Q-Q plot look like if the variables are normal?
qqnorm(normal)
qqline(normal)
```



```r
# compare
qqnorm(boot.mean)
qqline(boot.mean)
```



Bonus: Some other tricks for bootstrap sampling.

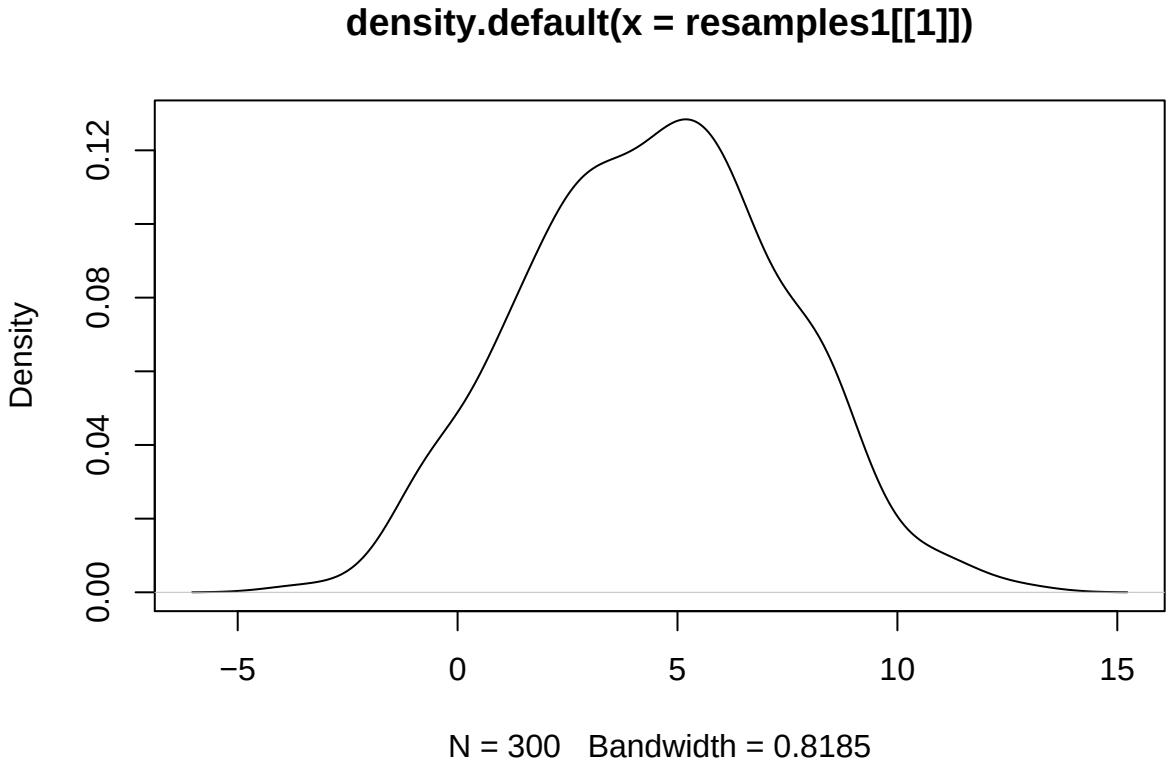
with replacement

```r
resamples1 <- lapply(1:20, function(i)
sample(data, replace = TRUE))
resamples1[1]
```

```
## [[1]]
##   [1]  3.88283801  2.66187253  7.71991936  6.12449155  5.11481566
##   [6]  4.76496318  0.44413509  8.35258664  3.88283801  2.55675014
##  [11]  1.28056871  1.84456540  6.69559812  1.36058122  5.62775723
##  [16]  5.75075731  6.08373189  6.25979629  8.95937762  0.37794834
##  [21]  4.93656248  8.16764484  7.05286752  5.53541149  8.05327925
##  [26]  3.60480767  4.10309072  1.87205571  6.42771113 -0.60614339
##  [31]  1.15139832 -0.22260122  7.42513996  4.38672617  7.40806835
##  [36]  6.78020653  6.66521680 -0.73928806  4.80363241  2.74275063
##  [41]  4.15374232 -0.27330237  0.48135906  4.76496318  1.06404443
##  [46]  3.59009684  3.58900640  6.29441159  5.74245276  7.37399958
##  [51]  1.89390294  7.37399958 -0.96444094  0.69326382  3.74357351
##  [56]  0.87136766  4.86736681  5.35045269  4.50369838  1.84456540
##  [61] -0.96444094  3.79047432  7.18330168  3.05009469  2.35427825
##  [66]  1.96503826  8.22674757  5.36070562  4.70257858  2.20110284
##  [71]  5.75075731  6.90352470 -0.60511736  1.89648306  3.88283801
##  [76]  4.39206918  3.16718781 11.16281890  2.47175723  7.85618016
```

```
##  [81]  4.86736681  4.86736681  1.96503826  3.88283801  1.89648306
##  [86]  5.14044174  7.01961675  3.49222427  8.59361684  8.59361684
##  [91]  6.17160411  2.51354959  1.28056871  5.14044174  7.01961675
##  [96]  2.94249672  5.38066183  7.99414966  0.74358176  5.97246455
## [101]  3.29587059  4.80363241  3.42772358  8.24885964  9.48286929
## [106]  4.15374232  0.91713168  2.75263345  7.97233985  5.89485110
## [111]  4.80363241  2.20110284  2.75263345  5.54799324  2.51354959
## [116]  2.53728921  0.69326382  6.90352470  3.16718781  6.37859234
## [121]  6.27266770  5.76037057  8.59361684  6.27266770  3.79047432
## [126]  1.98949348  2.47175723  1.95064606  7.17406043  3.49222427
## [131]  1.25458756  8.16764484 -3.57936444  3.69171070  2.55675014
## [136]  3.87639841  1.06404443  1.38972694  5.14044174  8.89253606
## [141]  5.19055296  6.50270470  3.87639841  5.14044174  5.90588029
## [146] 11.53783149  3.79047432  8.31085135  2.59601516  8.31085135
## [151]  4.38672617  3.74357351  6.35114913  4.76004084  5.14044174
## [156]  2.74275063  4.37546512  7.91535343 -1.39355916  3.50960509
## [161]  2.36305396  2.63213081  5.56630262 -0.01351180  1.00873040
## [166]  8.34291759  1.28251036  8.83337145 -0.96444094  0.46797974
## [171]  1.36058122  1.84456540  4.42107801  4.38672617  6.42771113
## [176]  3.26944249 -0.82405688  9.82457351  2.79849349  3.59009684
## [181]  2.44775359  3.92805914  5.32527549  4.80363241  1.41041592
## [186]  5.90588029  6.78020653  6.35379901  7.42513996  0.69326382
## [191]  1.25458756  4.76496318  6.13510017  6.17160411 10.73360845
## [196]  8.83337145  6.07200864  6.30500444  1.25458756  4.70257858
## [201]  3.42772358  9.84228225  3.45275645 -0.96444094  4.47508303
## [206]  7.72880253  5.35045269  1.00873040  5.85617360  2.66187253
## [211]  5.32527549  2.44775359  5.38823329  4.77480199  7.74963932
## [216] 12.77137254  5.14044174  7.11208907  6.57467375  1.36058122
## [221] 10.73360845  8.35258664  3.01180919  1.28251036  6.22027557
## [226] 11.16281890  1.41041592  1.89390294  4.87574195  6.13510017
## [231]  2.47175723 -0.01351180  4.19138261 -0.61408362  6.78020653
## [236]  3.47900693  5.74245276  3.42772358  8.02555318  2.67032635
## [241]  5.79195294 -1.39564737  3.03613066  2.54778350  5.14044174
## [246]  4.37546512  8.89253606 -0.64749855  2.75263345  6.73639942
## [251]  9.04507138  2.25597659  0.03704871  5.54799324  4.82018777
## [256]  3.36061202  9.04507138 -2.19986580  6.24765329  4.40629165
## [261]  8.40449085  0.41296519  8.31085135  2.72772481  9.48286929
## [266]  5.14044174  6.22749265  7.03488198  7.10573181  3.50960509
## [271]  4.36390778  5.52386861  7.71991936  8.49798052  4.82018777
## [276]  2.75263345  8.24885964  5.62775723  3.16718781  3.55763239
## [281]  9.93421015  1.84456540 -0.64749855  3.03613066 -0.55451073
## [286]  6.03159952  4.39206918  7.86216994  3.50960509  4.98513076
## [291]  5.00923230  6.24765329  5.29924263  5.74245276  6.23902497
## [296]  8.44782514  0.48135906  7.57918633  2.65876085 -0.27330237
```

```
plot(density(resamples1[[1]]))
```

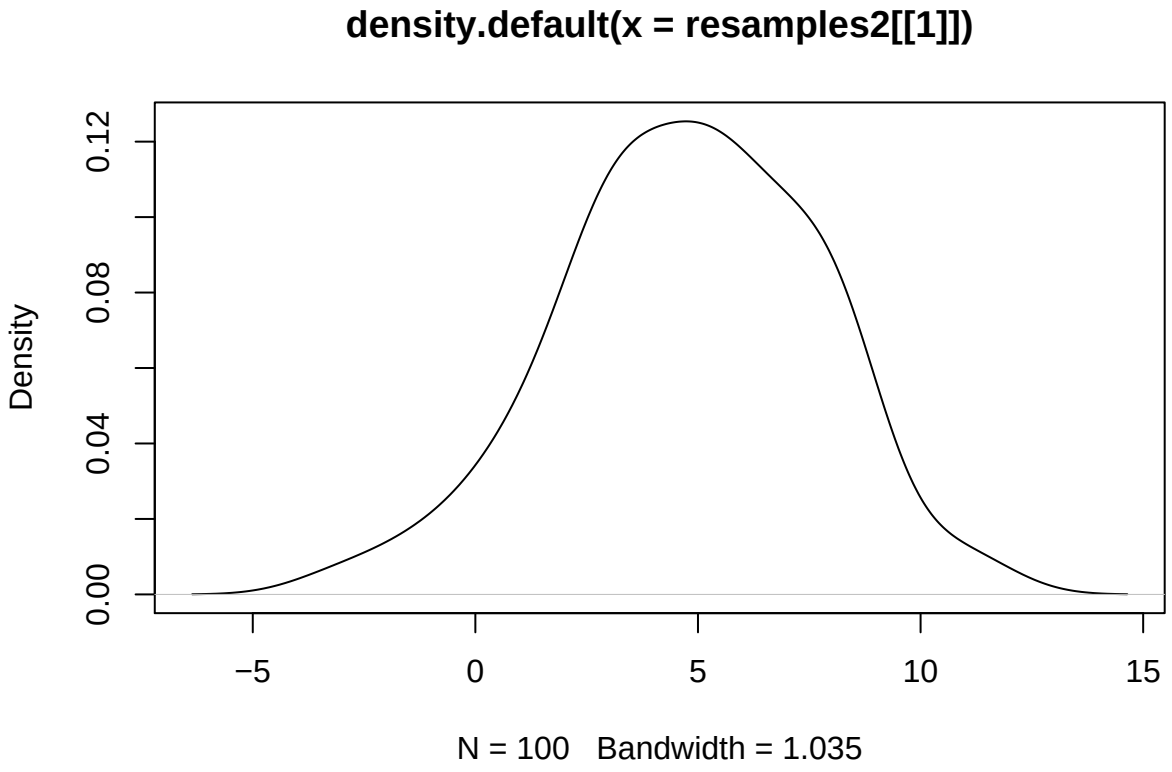


without replacement

```
resamples2 <- lapply(1:20, function(i)
sample(data, 100, replace = FALSE))
resamples2[1]
```

```
## [[1]]
##   [1]  5.3521438 10.4867978  7.4080683  5.3882333  6.2390250  3.5096051
##   [7]  5.3252755  8.0532792  3.3541023  1.7670867  4.7600408  8.3429176
##  [13]  5.2836069  5.8561736  6.7070527  8.3108514  4.8036324  6.0837319
##  [19]  8.2583436 -2.1998658 -1.3956474  4.8201878  3.2694425  7.1833017
##  [25]  6.2465364  1.8720557  6.2084730  1.8445654  3.7435735  8.9748147
##  [31]  5.7980585  4.9365625  2.3294929  7.8561802  6.7363994  7.1972656
##  [37] 10.8209844  5.1148157  0.5790077  2.5726200  3.4277236  7.9723399
##  [43]  3.0361307 -0.4079297  3.7937071  0.9817580  7.0416406 11.5378315
##  [49]  4.3182034  2.6618725  1.8964831  0.9171317  3.5890064  6.3511491
##  [55]  5.5354115  4.3639078  7.1057318  5.0772371 -0.9644409  5.7603706
##  [61]  4.3754651  8.0008577  7.9941497  3.0500947 -3.2556863  4.3920692
##  [67]  8.7778843  5.1404417  8.8342528  2.8950456  2.6321308  2.4477536
##  [73]  5.1190483  3.5457255  8.8925361  5.2992426  3.0118092  4.2629900
##  [79]  2.4717572 -0.1601112  8.1676448  1.2949199  6.6235517  0.7435818
##  [85]  4.8673668  6.3506138  2.9754508  0.3779483  6.0720086  7.0528675
##  [91]  1.8444633  7.7496393  2.9424967  3.6432674  8.0255532  3.7031211
##  [97]  7.0348820  8.3754058  4.3867262  3.4527565
```

```
plot(density(resamples2[[1]]))
```


density.default(x = resamples2[[1]])
Density
0.00
0.04
0.08
0.12
−5
0
5
10
15
N = 100 Bandwidth = 1.035