Bootstrap & Jackknife

Sahil Rangwala

12/13/2019

R Markdown

##Code for Bootstrap Method

```r
#Data is the mean annual temperature in degrees Fahrenheit in New Haven, Connecticut, from 1912 to 1971
#The data is a test data that is offered within Rstudio, hard coded the data due to technical difficult
#Calculating the probability of if the chosen mean annual temperature is less than the median of the da
#Should be around 50% probability, chose the median due to the fact that we know what the output should
#This would help hone in on making sure the code is correct for both the SimDesign and non SimDesign me

  data = c(49.9,52.3,49.4,51.1,49.4,47.9,49.8,50.9,49.3,51.9,50.8,49.6,49.3,50.6,48.4,50.7,50.9,50.6,51
  print(median(data))
```

```
## [1] 51.2
```

```r
  thetahat=length(data[data < 51.2]) / length(data) # calculate the theta hat
  B = 1000
  bootRet = numeric(B) # vector that houses the bootstrap results

  #Resampling portion
  for(b in 1:B) {
    sampleData = sample(data,length(data),replace=TRUE)

    #Seeing what samples are less than the median
    bootRet[b]= length(sampleData[sampleData < 51.2]) / length(sampleData)
  }

  #results of the bootstrap method
  mean_bootstrap=mean(bootRet)
  se_bootstrap=sd(bootRet)
  bias_bootstrap=mean(bootRet - thetahat)


print(mean_bootstrap)
```

```
## [1] 0.5007833
```

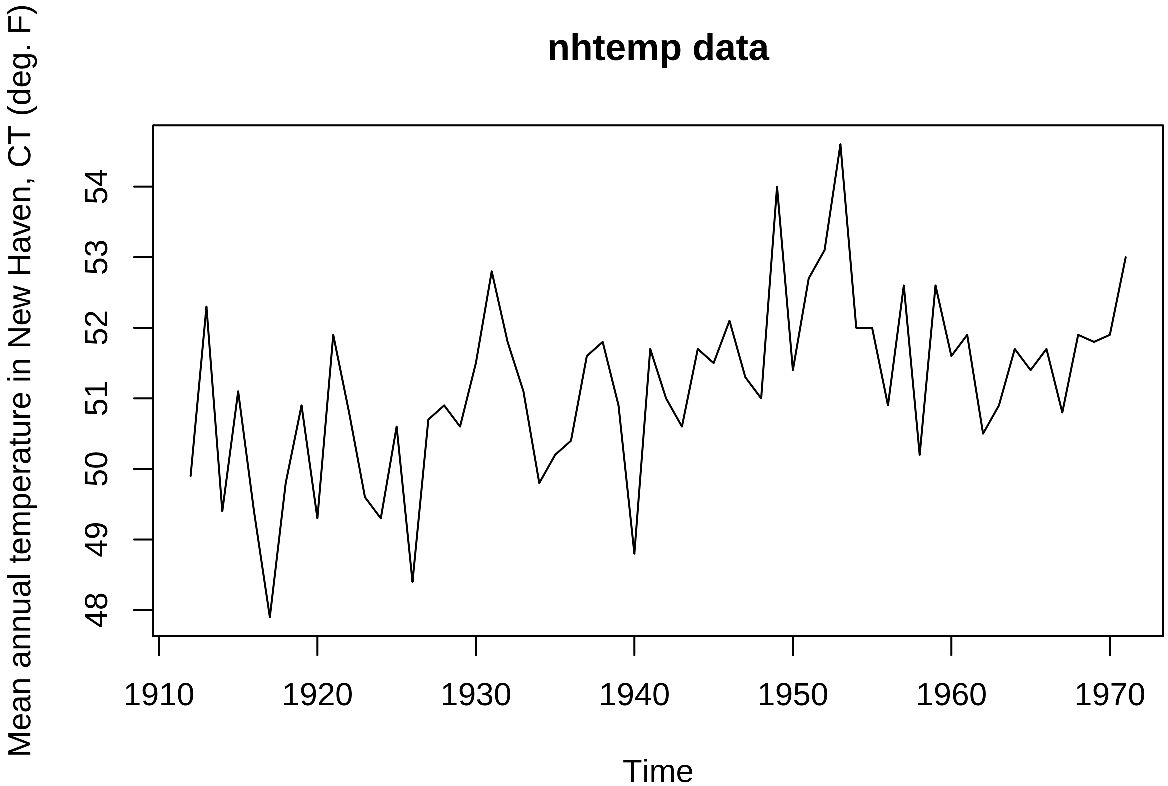
```r
print(se_bootstrap)
```

```
## [1] 0.06670584
```

```r
print(bias_bootstrap)
```

```
## [1] 0.0007833333
```

```r
#The data chosen
plot(nhtemp, main = "nhtemp data",
  ylab = "Mean annual temperature in New Haven, CT (deg. F)")
```



##Code for SimDesign Bootstrap Method

```
library(SimDesign)

data = c(49.9,52.3,49.4,51.1,49.4,47.9,49.8,50.9,49.3,51.9,50.8,49.6,49.3,50.6,48.4,50.7,50.9,50.6,51.5


thetaHat=length(data[data < 51.2]) / length(data) #Same method to calculate thetahat

Design <- expand.grid(condition = c(1)) #Sample chosen is already intialized

Generate <- function(condition, fixed_objects = NULL) {
    dat <- data
    return (dat)
}

 Analyse <- function(condition, dat, fixed_objects = NULL) {
    sampleChosen <- sample(dat,length(dat),replace=TRUE)
    ret <- c(xmean = length(sampleChosen[sampleChosen < 51.2]) / length(sampleChosen)) # calculating th
    return (ret)
}

Summarise <- function(condition, results, fixed_objects = NULL) {
    ret <- c(mean_Boot = mean(results[,1]), standardErrBoot = sd(results[,1]), biasBoot = mean(results[
    return (ret)
}

simBootRes <- runSimulation(design=Design, replications=1000, generate=Generate,
                         analyse=Analyse, summarise=Summarise) # replicating the simulation 1000
```

```
##
##
Design row: 1/1;   Started: Fri Dec 13 17:08:35 2019;   Total elapsed time: 0.00s
```

```
##
## Simulation complete. Total execution time: 0.34s
```

```r
print(simBootRes) # the results of the SimDesign implementation
```

```
##   condition mean_Boot standardErrBoot biasBoot REPLICATIONS SIM_TIME
## 1         1   0.50075      0.06322659  0.00075         1000    0.34s
##                   COMPLETED       SEED
## 1 Fri Dec 13 17:08:35 2019 1452142966
```

```r
data = c(49.9 ,52.3 ,49.4 ,51.1 ,49.4 ,47.9 ,49.8 ,50.9 ,49.3 ,51.9 ,50.8 ,49.6 ,49.3 ,50.6 ,48.4 ,50.7
```

Used http://philchalmers.github.io/SimDesign/html/11-Parametric-bootstrap.html to understand general structure

##Code for Jackknife Method

```r
data = c(49.9,52.3,49.4,51.1,49.4,47.9,49.8,50.9,49.3,51.9,50.8,49.6,49.3,50.6,48.4,50.7,50.9,50.6,51.5

  thetaHat = length(data[data<51.2])/length(data) # calculation for theta hat
  n = length(data)

  jackVal = numeric(n) # vector to save the results
  for (i in 1:n) {
    leftOut = data[-i] # leaving out the one specific data sample

    #Calculate the jackknife estimate for the specific sample
    jackVal[i] = length(leftOut [leftOut < 51.2]) / length(leftOut)
  }

  #results of the Jackknife method
  meanJack = mean(jackVal)
  sumsq = sum((jackVal - mean(jackVal))^2)
  standardErrorJack=sqrt((n - 1) / n) * sqrt(sumsq)
  biasJack=(n - 1) * (mean(jackVal) - thetaHat)

  #return in a form of vector

print(meanJack)
```

```
## [1] 0.5
```

```r
print(standardErrorJack)
```

```
## [1] 0.06509446
```

```r
print(biasJack)
```

```
## [1] 0
```

##Code for SimDesign Jackknife Method

```r
library(SimDesign)

data = c(49.9,52.3,49.4,51.1,49.4,47.9,49.8,50.9,49.3,51.9,50.8,49.6,49.3,50.6,48.4,50.7,50.9,50.6,51.5
```

```r
Design <- expand.grid(condition = c(1)) # don't need design because the samples are initialized

Generate <- function(condition, fixed_objects = NULL) {
    dat <- data
    dat
}

Analyse <- function(condition, dat, fixed_objects = NULL) {
    thetahat = length(dat[dat < 51.2]) / length(dat) # calculate the theta hat
    n = length(data)

    jackVal = numeric(n) # vector to save the results
    for (i in 1:n) {
      leftOut = data[-i] # leaving out the one specific data sample

      #Calculate the jackknife estimate for the specific sample
      jackVal[i] = length(leftOut [leftOut < 51.2]) / length(leftOut)
    }

    #results of the Jackknife method
    meanJack = mean(jackVal)
    sumsq = sum((jackVal - mean(jackVal))^2)
    standardErrorJack=sqrt((n - 1) / n) * sqrt(sumsq)
    biasJack=(n - 1) * (mean(jackVal) - thetaHat)



    ret <- c(meanJackknife = meanJack, standardErrorJackknife = standardErrorJack, BiasJackknife = bias
    return(ret)
}

Summarise <- function(condition, results, fixed_objects = NULL) {
    ret <- c(meanJackknife = mean(results[,1]),
             standardErrorJackknife = mean(results[,2]),
             BiasJackknife = mean(results[,3]))
    ret
}

SimdDesignResults <- runSimulation(design=Design, replications=1000, generate=Generate,
                         analyse=Analyse, summarise=Summarise)
```

```
##
##
Design row: 1/1;   Started: Fri Dec 13 17:08:35 2019;   Total elapsed time: 0.00s
##
## Simulation complete. Total execution time: 0.92s
```

```r
print(SimdDesignResults)
```

```
##   condition meanJackknife standardErrorJackknife BiasJackknife
## 1         1           0.5             0.06509446             0
##   REPLICATIONS SIM_TIME                COMPLETED       SEED
## 1         1000    0.92s Fri Dec 13 17:08:36 2019 2048925918
```

##Importance Sampling

```r
#initialize variables
m <- 10000
theta.hat <- se <- numeric(5)
#create function to represent equation
g <- function(x) {
exp(-x - log(1+x^2)) * (x > 0) * (x < 1)
}
#generate random uniform variables
u <- runif(m)
#Find importance function and calculate x
x <- - log(1 - u * (1 - exp(-1)))
#Find Ratio of functions
fg <- g(x) / (exp(-x) / (1 - exp(-1)))
#take mean of result
theta.hat <- mean(fg)
#calculate standard error
se <- sd(fg)
theta.hat
```

```
## [1] 0.5246061
```

```r
se
```

```
## [1] 0.09718451
```

##SimDesign Importance Sampling

```r
#Set Sample Size parameter for simulation design
Design <- data.frame(sample_size = c(100, 1000, 10000))

#Function used to generate data from uniform distribution
Generate <- function(condition, fixed_objects = NULL) {
    N <- condition$sample_size
    dat <- runif(N)
    dat
}

#Calculates Estimates
Analyse <- function(condition, dat, fixed_objects = NULL) {
    u <- dat
    #Find importance function and calculate x
    x <- - log(1 - u * (1 - exp(-1)))
    #Find Ratio of functions
    fg <- g(x) / (exp(-x) / (1 - exp(-1)))
    #take mean of result
    theta.hat <- mean(fg)
    se <- sd(fg)
    ret <- c(theta.hat = theta.hat,standardError=se)
    ret
}

#Caculates mean of estimates and standard error
Summarise <- function(condition, results, fixed_objects = NULL) {
    se=sd(results)
    ret <- c(theta.hat=mean((results[,1])), standardError = mean(results[,2]))
```

```r
    ret
}

#Simulation is run 1000 times
results_sd <- runSimulation(design=Design, replications=1000, generate=Generate,
                            analyse=Analyse, summarise=Summarise)
```

```
##
##
Design row: 1/3;   Started: Fri Dec 13 17:08:36 2019;   Total elapsed time: 0.00s
##
##
Design row: 2/3;   Started: Fri Dec 13 17:08:36 2019;   Total elapsed time: 0.31s
##
##
Design row: 3/3;   Started: Fri Dec 13 17:08:37 2019;   Total elapsed time: 0.81s
##
## Simulation complete. Total execution time: 2.85s
```

```r
results_sd
```

```
##   sample_size theta.hat standardError REPLICATIONS SIM_TIME
## 1         100 0.5250158    0.09660182         1000    0.31s
## 2        1000 0.5247865    0.09689087         1000    0.51s
## 3       10000 0.5248178    0.09685263         1000    2.04s
##                  COMPLETED       SEED
## 1 Fri Dec 13 17:08:36 2019  113459029
## 2 Fri Dec 13 17:08:37 2019 1587826100
## 3 Fri Dec 13 17:08:39 2019  346825591
```

##Monte Carlo Integration

```r
#function to generate monte carlo estimates
 pimc=function(a,b){
  m <- 10000
  #generate uniform random variables
  u <- runif(m,a,b)
  #function from integral to be estimated
  g=4*sqrt(1-u^2)
  #find estimate using mean function and taking into account integral limits
  theta.hat=mean(g)*(b-a)
  se=sd(g)
  return(c(theta.hat=theta.hat,sd=se))
 }
a=c(0,0,0)
b=c(1,0.5,.25)
theta.hat=numeric(3)
se=numeric(3)
for(i in 1:length(a)){
  print(pimc(a[i],b[i]))
}
```

```
## theta.hat        sd
## 3.1451549 0.8964739
## theta.hat        sd
## 1.9137549 0.1579013
```

```
##  theta.hat         sd
## 0.98956111 0.03764206
```

##SimDesign Monte Carlo Integration

```
#Set Parameters for simulation design
Design <- data.frame(sample_size = c(10000, 10000, 10000),
                     a = c(0, 0, 0),
                     b = c(1, 0.5, 0.25))

#Function used to generate data from uniform distribution with given parameters
Generate <- function(condition, fixed_objects = NULL) {
    N <- condition$sample_size
    ll <- condition$a
    ul <- condition$b
    dat <- runif(N, ll, ul)
    dat
}

#Calculates Estimates
Analyse <- function(condition, dat, fixed_objects = NULL) {
    u <- dat
    #function from integral to be estimated
    g=4*sqrt(1-u^2)
    #find estimate using mean function and taking into account integral limits
    theta.hat <- mean(g) * (condition$b-condition$a)
    ret <- c(theta_hat = theta.hat,se=sd(g))
    ret
}

#Caculates mean of estimates and standard error
Summarise <- function(condition, results, fixed_objects = NULL) {
    se = mean(results[,2])
    ret <- c(theta.hat = mean(results[,1]),standardError= se)
    ret
}

#Simulation is run 1000 times
results_sd <- runSimulation(design=Design, replications=1000, generate=Generate,
                         analyse=Analyse, summarise=Summarise)
```

```
##
##
Design row: 1/3;   Started: Fri Dec 13 17:08:39 2019;   Total elapsed time: 0.00s
##
##
Design row: 2/3;   Started: Fri Dec 13 17:08:40 2019;   Total elapsed time: 1.20s
##
##
Design row: 3/3;   Started: Fri Dec 13 17:08:41 2019;   Total elapsed time: 2.27s

##
## Simulation complete. Total execution time: 3.19s
```

```
results_sd
```

```
##   sample_size a    b theta.hat standardError REPLICATIONS SIM_TIME
```

```
## 1       10000 0    1 3.1416441    0.89273105         1000    1.20s
## 2       10000 0  0.5 1.9132146    0.15803139         1000    1.08s
## 3       10000 0 0.25 0.9894844    0.03777168         1000    0.91s
##                  COMPLETED       SEED
## 1 Fri Dec 13 17:08:40 2019  523686787
## 2 Fri Dec 13 17:08:41 2019  394522631
## 3 Fri Dec 13 17:08:42 2019 1666278706
```