DATA VISUALIZATION AND MODELING

SIMULATION – 3rd EXERCISE SHEET (NON-PARAMETRIC BOOTSTRAP)

Class: Data Visualization and Modeling **Professor:** Pere Puig

Students: Jamia Begum **NIU:** 1676891

Pedro Paixão Borges **NIU:** 1676893

1. **We're looking at the diversity of animal species in a particular forest. We've recorded the following: 13 animals in Species a, 40 animals in Species b, 9 animals in Species c, 20 animals in Species d, and 18 animals in Species f. Compute the Simpson's diversity index d and provide also a bootstrap estimate of d, its standard error and a 95% confidence interval.**

Species	**a**	**b**	**c**	**d**	**f**
No. of Animals	13	40	9	20	18

Answer:

Simpson's Diversity Index (SDI) measures the diversity of species in a community. It is denoted by D and computed as

$$D = 1 - \frac{\sum n_i(n_i - 1)}{N(N-1)}$$

where n_i represents the number of individuals of each species and N the total number of individuals of all species.

The value for Simpson's Diversity Index (D) ranges between 0 and 1, and the higher the value, the higher the diversity is. We compute D as follows:

Species	No. of individuals (n_i)	$(n_i - 1)$	$n_i(n_i - 1)$
a	13	12	156
b	40	39	1560
c	9	8	72
d	20	19	380
f	18	17	306
	N = 100		$\sum n_i(n_i - 1)$=2474

Therefore,

$$D = 1 - \frac{2474}{100(100-1)} = 0.75010101 \ldots \approx 0.75$$

We can compute D equally using R:

```r
n<-c(13,40,9,20,18) #number of individuals in each species
#create Function to calculate Simpson's diversity index (D)
SimpsonIndex <- function(n) {
  N <- sum(n)
  D<-1-sum(n*(n-1)/(N*(N-1))) #D=1-(Σn(n-1))/(N(N-1))
  return(D)
}
SimpsonIndex(n)
```

```
[1] 0.750101
```

Now, we proceed similarly to obtain a bootstrap estimate of D, its standard error and a 95% confidence interval. In this context, we use the following code in R:

```r
n <- c(13,40,9,20,18) # Number of individuals in each species

# Create function to calculate Simpson's diversity index (D)

SimpsonIndex <- function(n) {
  N <- sum(n)
  D <- 1-sum(n*(n-1)/(N*(N-1))) # D = 1-(Σn(n-1))/(N(N-1))
  return(D)
}
SimpsonIndex(n)

# Bootstrap estimation of D
l <- length(n)

# Set number of bootstrap samples
nsim <- 1000
stat <- numeric(nsim ) #create a vector in which to store the results

# Set up a loop to generate a series of bootstrap samples

for (i in 1:nsim){
     nB= sample (n , l, replace =T)
     stat[i] = SimpsonIndex(nB) # Calculate D for all bootstrap samples
     }

hist(stat)

# Bootstrap estimate of D
DB<- mean(stat)
DB


# Estimated standard error:
SE<-sd(stat)
SE

# 95% Confidence Interval:
quantile(stat,c(0.025,0.975))
```

Then, our bootstrap estimate of D is 0.7660597 with a standard error of 0.02642658 and [0.7199017, 0.8056669] as its 95% confidence interval.

```
> # Bootstrap estimate of D
> DB<- mean(stat)
> DB
[1] 0.7660597
> # Estimated standard error:
> SE<-sd(stat)
> SE
[1] 0.02642658
```

```
> # 95% Confidence Interval:
> quantile(stat,c(0.025,0.975))
     2.5%     97.5%
0.7199017 0.8056669
```

2. We have a sample of 6 cars in a second-hand market, of the same model and year, with the following prices and total km:

Km	**37388**	**44758**	**45833**	**30862**	**31705**	**34010**
Price	14636	14122	14016	15590	15568	14718

Calculate the 95% bootstrap CI (three methods) of the second-hand price for a car (same model and year) with 50000 km.

We start by defining the general data that is going to be used for the problem. In our case, we will predict the price of the second-hand car using a linear regression. Therefore, the intercept and slope parameters are going to compose the parameter $\theta = (\theta_{intercept}, \theta_{slope})$ that we want to estimate. We start by computing it directly from the data, using a MSE strategy, and from there we obtain an initial estimate for the price of a car with 50000km.

```
km    <- c(37388,44758,45833,30862,31705,34010)
price <- c(14636,14122,14016,15590,15568,14718)
df <- data.frame(km, price) # y is price, x is km
n  <- length(price)
fit<-lm(df[,2]~df[,1])
s<-summary(fit)

# Obtaining intercept and slope directly from data
theta<-c(s$coefficients[1], s$coefficients[2])
intercept <-theta[1]
slope <- theta[2]

# Standard deviation of each one of the components
sdtheta<-c(s$coefficients[3], s$coefficients[4])

# Price estimation
fitb <- lm(price~km, data=df)
pred <- predict(fitb, data.frame(km = c(50000)), se.fit=TRUE)
initp <- price$fit        # Initial price prediction
sd_initp <- price$se.fit  # Standard deviation for initial price prediction
```

The values of our initial estimates are

```
> theta
[1]  1.845807e+04 -9.840931e-02

> initp
       1
13537.6
```

Given that, the three methods that can be used to calculate confidence intervals via bootstrap are:

- **Percentile/Quartile method**

  Suppose that we have data given as $\boldsymbol{x} = (x_1, x_2, \dots, x_n)$, for which we obtain a parameter estimation $\hat{\theta} = s(\boldsymbol{x})$. We resample B times, obtaining new samples $\boldsymbol{x_1^*}, \boldsymbol{x_2^*}, \dots, \boldsymbol{x_B^*}$. For each one of this, we get resample statistics $\widehat{\theta_b^*} = s(\boldsymbol{x}_b^*)$ and then we sort them. $LCL = \widehat{\theta_{(v_1)}^*}$ and UCL = $\widehat{\theta_{(v_2)}^*}$, where v_1 and v_2 are the orders of the $\frac{\alpha}{2}$ and $1 - \frac{\alpha}{2}$ quantiles.
  This type of method is good when the estimate is symmetrically distributed about the true value of the parameter and the tails of the estimate's distribution drop off rapidly to zero.

  This is done via the following code:

```
nb <- 1000
y <- seq(1:n) #
pb <- numeric(nb) # Create a vector to store predictions

# Quantile bootstrap to estimate the price for the car with 50000km

set.seed(123)

for(i in 1:nb){
  yb <- sample(y, n, replace=T) # Sample 1000 times values index from 1 to 6
  dfb = data.frame(km = df[yb, 1], price = df[yb, 2])
  fitb <- lm(price~km, data = dfb)
  p <- predict(fitb, data.frame(km = c(50000)))
  pb[i] <- p
}
# Bootstrap estimate of the price

mean(pb)

# 95% confidence interval:

ic <- quantile(pb,c(0.025,0.975))
```

Obtaining as results

```
> mean(pb)
[1] 13442.71

> ic
    2.5%    97.5%
12217.61 13806.00
```

- **Simple method**

Suppose that we have data given as $\boldsymbol{x} = (x_1, x_2, \dots, x_n)$, for which we obtain a parameter estimation $\hat{\theta} = s(\boldsymbol{x})$. We resample B times, obtaining new samples $\boldsymbol{x_1^*}, \boldsymbol{x_2^*}, \dots, \boldsymbol{x_B^*}$. For each one of this, we get resample statistics $\widehat{\theta_b^*} = s(\boldsymbol{x_b^*})$ and then we sort them. $LCL = 2\hat{\theta} - \widehat{\theta^*_{(v_2)}}$ and $UCL = 2\hat{\theta} - \widehat{\theta^*_{(v_1)}}$, where v_1 and v_2 are the orders of the $\frac{\alpha}{2}$ and $1 - \frac{\alpha}{2}$ quantiles.

In order to compute the simulation in this case, we use the following code:

```
# The simple method to calculate 95% Confidence Interval:

lb <- 2*initp - quantile (pb,0.975)
ub <- 2*initp - quantile (pb,0.025)
ic <- c(lb,ub)
names(ic) <- c("2.5%", "97.5%")
ic
```

The results are as follows:

```
> ic
     2.5%     97.5%
13269.21 14857.60
```

- **Bootstrap-t method**

  As assumptions, we consider that we have consistent estimates of θ_0 and its standard error at hand. Besides, we consider that the asymptotic distribution of the t-statistic is the standard normal, i.e. $t = \frac{\hat{\theta}-\theta_0}{\widehat{se}(\hat{\theta})} \rightarrow^d N(0,1)$.
  We start by drawing B independent bootstrap samples of size n from the original sample. We estimate the t-value for each sample by

$$t_b^* = \frac{\widehat{\theta_b^*} - \hat{\theta}}{\widehat{se}_b^*(\hat{\theta})} \quad for\ b = 1, \dots, B$$

  where $\widehat{\theta_b^*}$ and $\widehat{se}_b^*(\hat{\theta})$ are estimates of the parameter θ and its standard error using bootstrap sample.

  Afterwards, we calculate $t_{\alpha/2}$ and $t_{1-\alpha/2}$ by means of the quantiles of t_b^*. Finally, the confidence interval is given as

$$[\hat{\theta} + t_{\frac{\alpha}{2}} \cdot \widehat{se}(\hat{\theta}), \hat{\theta} + t_{1-\frac{\alpha}{2}} \cdot \widehat{se}(\hat{\theta})]$$

This is performed via the following code:

```
pb <- list()
sdb <- list()

for (i in 1:nb){
  yb <- sample(y,n,replace =T)
  df_yb = data.frame(price = df[yb,2], km = df[yb,1])
  fitb <- lm(price~km, data=df_yb)
  price <- predict(fitb, data.frame(km = c(50000)), se.fit=TRUE)
  pb[[i]] <- price$fit
  sdb[[i]] <- price$se.fit

}

pb <- data.frame(price = unlist(pb))
sdb <- data.frame(sd = unlist(sdb))

# Estimating t value for each sample

tb_price <- (pb$price - initp)/sdb
tb <- data.frame(tb_price)
names(tb) <- c("price")

tb_pl <- quantile(tb$price,0.025) # Lower CI for tb
tb_pu <- quantile(tb$price,0.975) # Upper CI for tb

pl <- initp + tb_pl * sdp # Lower CI for price
pu <- initp + tb_pu * sdp # Upper CI for price

ic <- c(pl,pu)
ic
```

And the confidence interval for the final price is given as follows:

```
> ic
       1        1
12707.32 16757.20
```

Another way of approaching the problem – constructing confidence intervals for the parameters

Instead of obtaining the price estimation directly, another possibility is simulating and obtaining confidence intervals for the parameters of the linear regression itself, i.e., for the slope and intercept rates. This is done as follows:

- **Percentile/Quartile method**

```
# Bootstraping

set.seed(1)
nb <- 1000
y <- seq (1,n)
interceptb <- numeric(nb)
slopeb <- numeric(nb)

for(i in 1:nb){
  yb <- sample( y,n,replace =T)
  fitb <- lm(df[yb,2]~df[yb,1])
  sb<-summary(fitb)
  interceptb[i] <- sb$coefficients[1]
  slopeb[i] <- sb$coefficients[2]
}

iu <- quantile(interceptb,0.975) # Upper CI for intercept
il <- quantile(interceptb,0.025) # Lower CI for intercept
su <- quantile(slopeb,0.975) # Upper CI for slope
sl <- quantile(slopeb,0.025) # Lower CI for slope
```

For the parameters CI, we have the following results.

```
> c(il,iu)
    2.5%    97.5%
13447.38 25939.36
> c(sl,su)
       2.5%       97.5%
-0.28474083  0.01166125
```

- **T-bootstrap**

We use the following code to perform the computations.

```
# Now, using the bootstrap-t method
thetab <- list()
sdb <- list()

for (i in 1:nb){
  yb <- sample(y,n,replace =T)
  fitb <- lm(df[yb,2]~df[yb,1])
  sb<-summary(fitb)
  theta <- c(sb$coefficients[1],sb$coefficients[2]) # Intercept and slope
  thetab[[i]] <- theta
  sd <- c(sb$coefficients[3], sb$coefficients[4]) # Standard deviations for bootstrap
  sdb[[i]] <- sd
}

thetab <- do.call(rbind.data.frame, thetab)
sdb <- do.call(rbind.data.frame, sdb)
names(thetab) <- c("intercept","slope")
names(sdb) <- c("intercept_sd","slope_sd")

# Estimating t value for each sample
tb_intercept <- (thetab$intercept - intercept)/sdb$intercept_sd
tb_slope <- (thetab$slope - slope)/sdb$slope_sd
tb <- data.frame(intercept = tb_intercept, slope = tb_slope)
```

We also add an extra line to remove outliers in tb (value bigger than 10^6), generated when the standard deviation for the slope is very close to zero. After that, we can finally obtain the confidence intervals for the parameters and use that to compute the confidence intervals for price.

```
tb_iu <- quantile(tb$intercept,0.975) # Upper CI for tb (intercept)
tb_il <- quantile(tb$intercept,0.025) # Lower CI for tb (intercept)
tb_su <- quantile(tb$slope,0.975)  # Upper CI for tb (slope)
tb_sl <- quantile(tb$slope,0.025) # Lower CI for tb (slope)

il <- theta[[1]] + tb_il * sdtheta[[1]]
iu <- theta[[1]] + tb_iu * sdtheta[[1]]
sl <- theta[[2]] + tb_sl * sdtheta[[2]]
su <- theta[[2]] + tb_su * sdtheta[[2]]
```

As result, we have for the intercept rate the following confidence interval:

```
> c(il, iu)
    2.5%    97.5%
11599.03 25001.86
```

And for the slope, the confidence interval:

```
> c(sl, su)
       2.5%        97.5%
-0.27019062  0.06798036
```