

STA380 Practice Problems for Quiz 5

These problems are not to be handed in, but they are for extra practice for students to be prepared for the quiz.

1. Install the [MEMSS](#) package (`install.packages(MEMSS)`).

Consider the `RatPupWeight` dataset from the package.

You can easily load this using `data(RatPupWeight, package = "MEMSS")`.

You can run the `??MEMSS::RatPupWeight` command to learn a bit about the variables of the dataset.

- (a) Using nonparametric bootstrapping, estimate the mean of the rat's weights. Use 200 replicates.
- (b) Compute the bootstrap estimate of the standard error of the bootstrap sample mean. That is,

$$SE(\bar{x}^*) = \sqrt{\frac{\sum_{b=1}^B [(\bar{x}^*)^{(b)} - \bar{x}']^2}{B-1}}.$$

- (c) Compute the bootstrap estimate of bias of the bootstrap sample mean. That is,

$$Bias(\bar{x}^*) = \frac{1}{B} \sum_{b=1}^B (\bar{x}^*)^{(b)} - \bar{x}.$$

- (d) Provide a 95% bootstrap percentile confidence interval for the mean of the rat's weights.
- (e) We want to test whether the weights of the male and female pups come from the same distribution. Perform a permutation test for equal distributions by computing the Kolmogorov-Smirnov (K-S) Statistic. (You are free to use `ks.test()` instead of brute forcing like we did in lecture.) Feel free to use 999 replicates.

2. Consider the `Relaxin` dataset from the [MEMSS](#) package.

You can easily load this using `data(Relaxin, package = "MEMSS")`.

You can run the `??MEMSS::Relaxin` command to learn a bit about the variables of the dataset.

- (a) Using nonparametric bootstrapping, estimate the **standard error** of the glucose concentration levels. Use 200 replicates.
- (b) Using jackknife, estimate the **standard error** of the glucose concentration levels.
- (c) Using nonparametric bootstrapping, estimate the **bias of the variance estimator**, as described below, of the glucose concentration levels. The estimator is:

$$\widehat{\sigma^2} = \frac{1}{n} \sum_{i=1}^n (x_i - \bar{x})^2$$

Use 200 replicates.

- (d) Using jackknife, estimate the **bias of the variance estimator**, the one described in the last question, of the glucose concentration levels.
- (e) When it comes to parts (c) and (d), the results between the bootstrap and the jackknife differ somewhat significantly. Why do you think this is the case? *Hint: consider that there are cases where the Jackknife is unreliable, or the proofs done in lecture.*

3. Consider the `Milk` dataset from the [MEMSS](#) package.

You can easily load this using `data(Milk, package = "MEMSS")`.

You can run the `??MEMSS::Milk` command to learn a bit about the variables of the dataset.

- (a) Check the histogram of the protein levels (`hist(Milk$protein)`). Do you expect the jackknife method to work better here? Why or why not?
- (b) Using nonparametric bootstrapping, estimate the **bias of the variance estimator**, as described below, of the glucose concentration levels. The estimator is:

$$\widehat{\sigma^2} = \frac{1}{n} \sum_{i=1}^n (x_i - \bar{x})^2$$

Use 200 replicates.

- (c) Using jackknife, estimate the **bias of the variance estimator**, the one described in the last question, of the glucose concentration levels.

Below are the solutions.

1. If you want to see parts a), b), etc., see the comments on the following code.

```
#install.packages("MEMSS")
library("MEMSS")

data(RatPupWeight, package = "MEMSS")

# to read the documentation
??MEMSS::RatPupWeight

B <- 200
n <- nrow(RatPupWeight)
# this is to obtain which indices to sample from.
mat <- matrix(
  sample.int(n, size = n * B, replace = TRUE),
  nrow = B, ncol = n)
# now we want the matrix that contains the actual sampled weight value
weights <- matrix(RatPupWeight$weight[mat], nrow = B, ncol = n)
# computing the mean of each replicate
R <- rowMeans(weights)

# a) computing the bootstrap mean
mean(R)
# optional: comparing this to the sample mean
samp_mean <- mean(RatPupWeight$weight)
samp_mean

# b) computing the bootstrap standard error
sd(R)
# optional: comparing this to the sample standard deviation
sd(RatPupWeight$weight) / sqrt(n)

# c) computing the bootstrap bias
mean(R) - samp_mean

# computing the 95% bootstrap percentile confidence interval
# for the mean.

# d) we can be efficient and use what was previously computed.
alpha = 0.05
c(quantile(R, alpha/2), quantile(R, 1-alpha/2))

mean(RatPupWeight$weight)

# e)
f <- RatPupWeight[RatPupWeight$sex == "Female", ]
m <- RatPupWeight[RatPupWeight$sex == "Male", ]
D0 <- suppressWarnings(ks.test(f$weight, m$weight)$statistic)

R <- 999
D <- numeric(R)
for (i in 1:R) {
  k <- sample(1:n, size = n/2, replace = FALSE)
  xi <- RatPupWeight$weight[k]
  yi <- RatPupWeight$weight[-k]
  # the warning isn't applicable to us as it's for a different value than
  # the D statistic.
  # If this makes you uncomfortable, you can brute force the value for D
  # like we did in lecture.
  D[i] <- suppressWarnings(ks.test(xi, yi)$statistic)
}
```

```
# we need to include D0 as technically, it counts as a sample...
p <- mean(c(D0, D) >= D0)
p
```

2. `data(Relaxin, package = "MEMSS")`

```
??MEMSS::Relaxin

# Beginning with bootstrap method
B <- 200
n <- nrow(Relaxin)
mat <- matrix(
  sample.int(n, size = n * B, replace = TRUE),
  nrow = B, ncol = n)
rel_conc <- matrix(Relaxin$conc[mat], nrow = B, ncol = n)

# a) standard error
R <- rowMeans(rel_conc)
sd(R)

# Jackknife method,
jack_est <- numeric(n)
for(i in 1:n){
  new_samp <- Relaxin$conc[-i]
  jack_est[i] <- mean(new_samp)
}

# b)
sqrt((n-1)/n * sum((jack_est - mean(jack_est))^2))

# c)
boot_est <- (1/n) * rowSums((rel_conc - rowMeans(rel_conc))^2)
var_mle <- (1/n) * sum((Relaxin$conc - mean(Relaxin$conc))^2)

mean(boot_est) - var_mle

# d)
jack_est_bias <- numeric(n)
var_est <- (1/n) * sum((Relaxin$conc - mean(Relaxin$conc))^2)

for(i in 1:n){
  new_samp <- Relaxin$conc[-i]
  jack_est_bias[i] <- (1/(n-1)) * sum((new_samp - mean(new_samp))^2)
}

(n-1)*(mean(jack_est_bias) - var_est)
```

Try running `hist(Relaxin$conc)` and you'll see that the values are not Gaussian distributed, which is why the jackknife performs horribly.

3. Since the values are Gaussian distributed, we expect the jackknife to perform better.

```
data(Milk, package = "MEMSS")
# this one actually looks somewhat Gaussian distributed
hist(Milk$protein)

# Bootstrap version
B <- 200
n <- nrow(Milk)
mat <- matrix(
  sample.int(n, size = n * B, replace = TRUE),
  nrow = B, ncol = n)
```

```

milk_pro <- matrix(Milk$protein[mat], nrow = B, ncol = n)

boot_est <- (1/n) * rowSums((milk_pro - rowMeans(milk_pro))^2)
var_mle <- (1/n) * sum((Milk$protein - mean(Milk$protein))^2)

# a)
boot_ver <- mean(boot_est) - var_mle

# Jackknife version
jack_est_bias <- numeric(n)
var_est <- (1/n) * sum((Milk$protein - mean(Milk$protein))^2)

for(i in 1:n){
  new_samp <- Milk$protein[-i]
  jack_est_bias[i] <- (1/(n-1)) * sum((new_samp - mean(new_samp))^2)
}

jackknife_ver <- (n-1)*(mean(jack_est_bias) - var_est)

abs(boot_ver - jackknife_ver)

```