S3 Method Bootstrapping Exercise

Eliza Chai

05/11/2022

In this problem, you will construct an S3 method `bootstrap`, for both the class `numeric` and `stratified` (introduced in Lecture 5), with the following interface.

```r
bootstrap.my_class <- function(object, nboot, stat){... your code here ...}
```

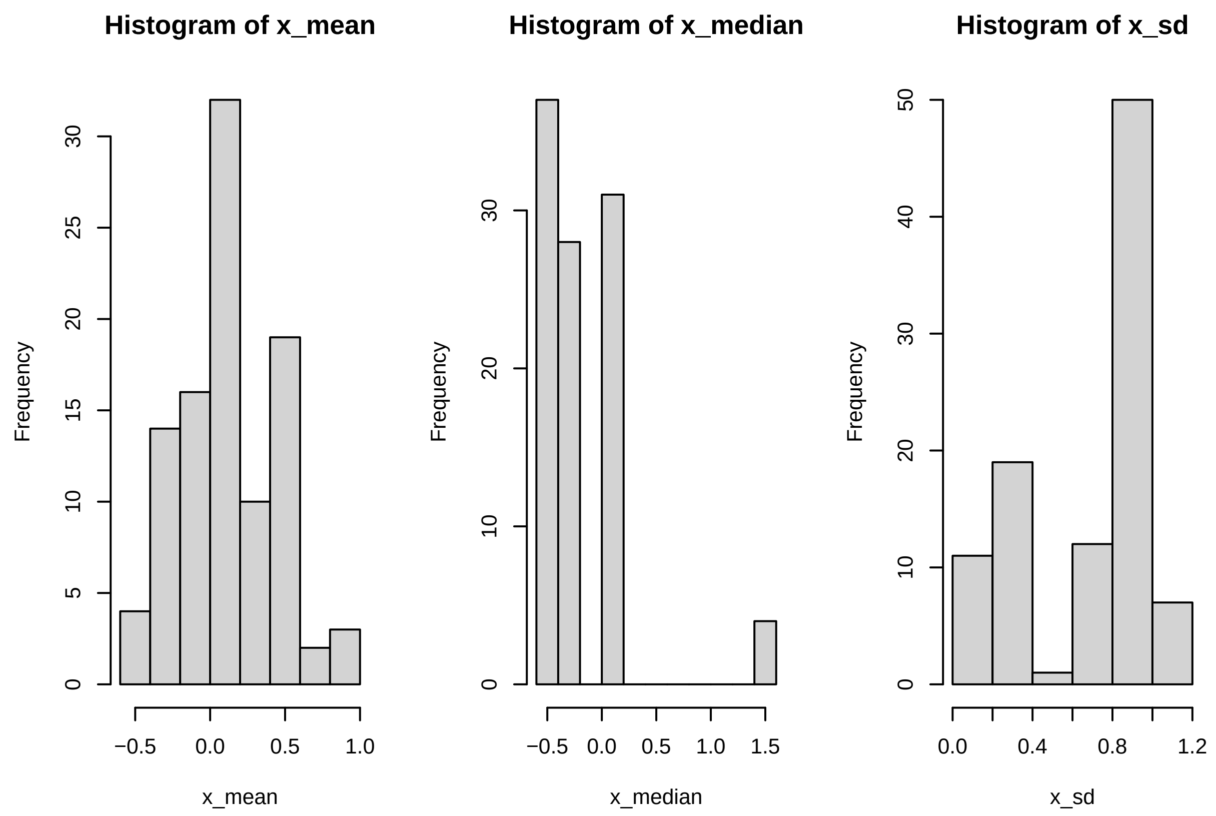
The function `bootstrap.my_class` will return the *evaluations* of the statistics (i.e., function) encoded in the function `stat` on each one of the bootstrapped vectors.

Illustrate the use of your `bootstrap` generic function on objects of the class `numeric` and `stratified` using the mean, the median, and the standard deviation as the statistics of interest (e.g. make a histogram with the evaluations of the statistics).

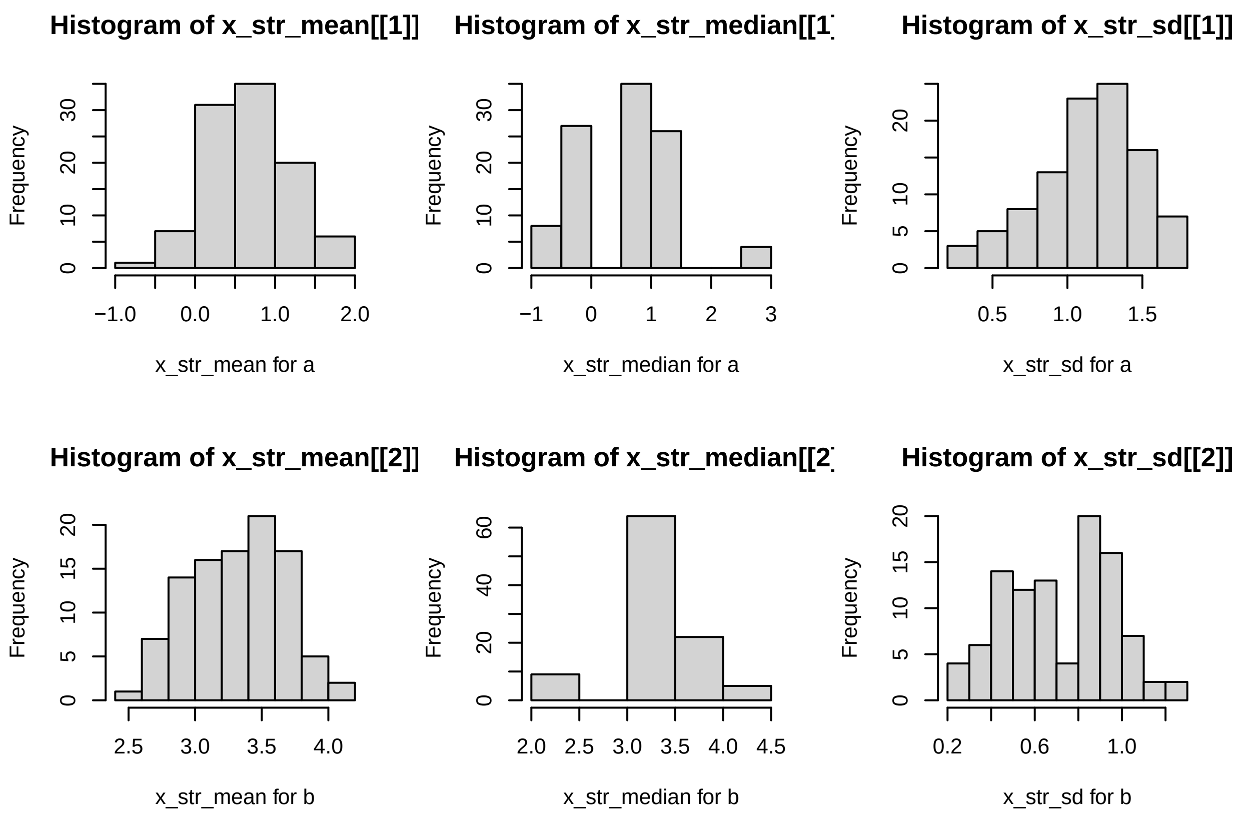
```r
## Constructor functions to make sure the objects is class: stratified
stratified <- function(y, strata) {
        if (!is.numeric(y)) stop("'y' must be numeric")
        if (!is.factor(strata)) stop("'strata' must be a factor")

        if (length(y) != length(strata)) stop("'y' and 'strata' must have equal length")

        structure(list(y=y, strata=strata), class = "stratified")
}

## Create a generic bootstrap function
bootstrap <- function(object, ...) UseMethod("bootstrap")

## Create a method for numeric vectors
bootstrap.numeric <- function(object, nboot, stat){
        if ( !is( object, "numeric") )
                stop( "bootstrap.numeric requires an object of class 'numeric'" )
        if ( nboot < 1 | is.infinite(nboot) )
                stop( "'nboot' should be a positive integer" )

        n <- length(object)

        boot_samp <- replicate(nboot, sample(object, size=n, replace=TRUE))

        colnames(boot_samp) <- paste("b", 1:nboot, sep="")

        boot_stat <- apply(boot_samp, 2, stat)

        return(boot_stat)
}
```

```r
## Create a method for stratified vectors
bootstrap.stratified <- function(object, nboot, stat){
        if ( !is( object, "stratified") )
                stop( "bootstrap.stratified requires an object of class 'stratified'" )
        if ( nboot < 1 | is.infinite(nboot) )
                stop( "'nboot' should be a positive integer" )

        tapply(object$y, object$strata, bootstrap.numeric, nboot, stat)
}


x <- rnorm(5)
x_mean <- bootstrap(x, 100, mean)
x_median <- bootstrap(x, 100, median)
x_sd <- bootstrap(x, 100, sd)

x_str <- stratified(y = c(rnorm(5), rnorm(5, 3)),
                          strata = factor(rep(c("a","b"), each=5)) )

x_str_mean <- bootstrap(x_str, 100, mean)
x_str_median <- bootstrap(x_str, 100, median)
x_str_sd <- bootstrap(x_str, 100, sd)
```

```r
## Create histograms
par(mfrow=c(1,3))
hist(x_mean)
hist(x_median)
hist(x_sd)
```



```
par(mfrow=c(2,3))
hist(x_str_mean[[1]], xlab= "x_str_mean for a")
hist(x_str_median[[1]], xlab= "x_str_median for a")
hist(x_str_sd[[1]], xlab= "x_str_sd for a")
hist(x_str_mean[[2]], xlab= "x_str_mean for b")
hist(x_str_median[[2]], xlab= "x_str_median for b")
hist(x_str_sd[[2]], xlab= "x_str_sd for b")
```



Generalize the methods `bootstrap` defined above to the case of an argument `stat` that is a function that can take additional arguments, e.g. a function that computes the **k**th moment. Test it.

```r
moment <- function(x, k)
{
(1/length(x))*sum((x-mean(x))^k)
}
```

```r
## Create a generic bootstrap function
bootstrap <- function(object, ...) UseMethod("bootstrap")
## Create a method for numeric vectors
bootstrap.numeric <- function(object, nboot, stat, k){
        if ( !is( object, "numeric") )
                stop( "bootstrap.numeric requires an object of class 'numeric'" )
        if ( nboot < 1 | is.infinite(nboot) )
                stop( "'nboot' should be a positive integer" )

        n <- length(object)

        boot_samp <- replicate(nboot, sample(object, size=n, replace=TRUE))

        colnames(boot_samp) <- paste("b", 1:nboot, sep="")

        boot_stat <- apply(boot_samp, 2, stat)

        stat <- function(x, ...){
                moment(...)
        }
        moment <- function(x, k)
        {
                (1/length(x))*sum((x-mean(x))^k)
        }
        return(boot_stat)
}


bootstrap.stratified <- function(object, nboot, stat){
        if ( !is( object, "stratified") )
                stop( "bootstrap.stratified requires an object of class 'stratified'" )
        if ( nboot < 1 | is.infinite(nboot) )
                stop( "'nboot' should be a positive integer" )

        tapply(object$y, object$strata, bootstrap.numeric, nboot, stat)
}
```