Bootstrapping

Makayla Tang

2022 May

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).

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 kth moment. Test it.

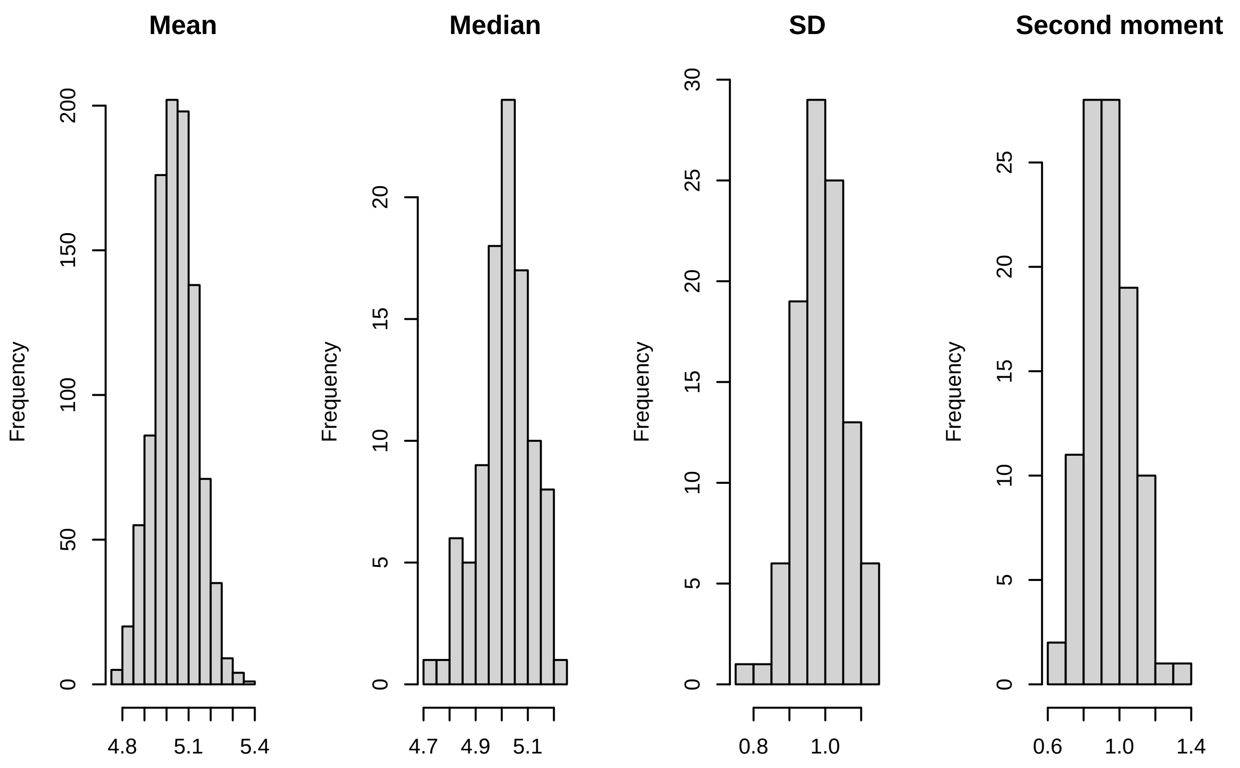
```r
bootstrap <- function(object, ...) UseMethod("bootstrap")
```

```r
bootstrap.numeric <- function(object, nboot, stat, ...){
        if (!is( object, "numeric"))
                stop( "bootstrap.numeric must be 'numeric'" )
        if (nboot < 1 | is.infinite(nboot))
                stop( "'nboot' should be a positive integer" )
        if (!is( stat, "function"))
                stop( "bootstrap.numeric requires 'stat' of class 'function'" )
        n <- length(object)
        purrr::map_dbl(seq(nboot), function(x) stat(sample(object, size=n, replace=TRUE), ...))
}
```

Visualization

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

x <- rnorm(100,5,1)
par(mfrow=c(1,4))
hist(bootstrap(x, 1000, mean), main="Mean", xlab="")
hist(bootstrap(x, 100, median), main="Median", xlab="")
hist(bootstrap(x, 100, sd), main="SD", xlab="")
hist(bootstrap(x, 100, moment, k = 2), main="Second moment", xlab="")
```



Constructor function for 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")
}
```

bootstrap method for *stratified* objects

```
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" )
        if ( !is( stat, "function") )
                stop( "bootstrap.numeric requires 'stat' of class 'function'" )

        stat_with_args = function(x) stat(x, ...)
        tapply(object$y, object$strata, bootstrap.numeric, nboot, stat_with_args)
}
```

Visualization

```
my_str_samp <- stratified(y = c(rgamma(50,3), rnorm(100, 30)),
                          strata = factor(c(rep("a",50),rep("b",100))))
bootstrap(my_str_samp, 10, mean)
```

```
## $a
##  [1] 2.391750 2.925965 2.664088 2.981416 2.574893 2.962874 2.612968 3.037312
##  [9] 2.904482 3.108343
##
## $b
##  [1] 30.09091 30.17349 29.98928 30.27921 30.19636 30.33566 30.13425 30.36530
##  [9] 30.08087 30.09921
```

```
bootstrap(my_str_samp, 10, median)
```

```
## $a
##  [1] 2.158740 2.206503 2.173276 2.168560 2.201787 2.169051 1.903355 2.173767
##  [9] 2.203292 2.173276
##
## $b
##  [1] 30.04945 30.15225 30.04945 30.18809 30.15225 29.96243 30.09193 30.13855
##  [9] 30.19018 30.13855
```

```
bootstrap(my_str_samp, 10, sd)
```

```
## $a
##  [1] 2.015179 1.925486 2.273661 2.632514 2.059128 2.010582 1.717821 2.103568
##  [9] 2.071740 1.856548
##
## $b
##  [1] 1.0440252 1.0967125 1.0582174 1.0447260 1.0310606 1.1402648 1.0261902
##  [8] 0.9587717 0.9981911 1.1057161
```

```
bootstrap(my_str_samp, 10, moment, 2)
```

```
## $a
##  [1] 3.501786 5.445695 4.534420 5.816643 4.628013 5.638926 4.547240 3.305014
##  [9] 3.408344 2.588797
##
## $b
##  [1] 1.0286598 0.8363398 1.0200678 0.9377765 0.9955167 1.0060387 1.1372711
##  [8] 0.9457625 1.1646599 0.8464499
```