Using BLBtools6520 for Scalable Bootstrap Analysis

Lehao Fu & Boyu Wang

Introduction

The `BLBtools6520` package provides tools to perform scalable bootstrap analyses using the Bag of Little Bootstraps (BLB) algorithm and its parallelized version, as well as a subsampling-based algorithm for comparison. These methods are designed for handling large datasets efficiently.

This vignette introduces the three main functions in the package:

1. `BLB`: The standard BLB implementation.
2. `ScaSub`: A subsampling-based method for comparison.
3. `BLBpar`: A parallelized version of BLB for faster computation.

Installation

To install the package, you can use the following command:

```
# Install devtools if not already installed
install.packages("devtools")

# Install BLBtools6520 from GitHub
devtools::install_github("FuLehao/BLBtools6520")
```

Load the package:

```
library(BLBtools6520)
```

Function Descriptions and Examples

1. `BLB`

The `BLB` function implements the Bag of Little Bootstraps algorithm for scalable resampling.

Parameters:

- `data`: A numeric vector of data values.
- `beta`: A numeric value determining the size of each subset, `b = n^beta`, where `n` is the length of the dataset.
- `s`: The number of subsets to generate.
- `r`: The number of resampling iterations for each subset.
- `u`: A function to calculate the point estimate (e.g., `mean`, `median`).
- `xi`: A function to compute a confidence interval or other summary from the resampling results.

```

data <- rnorm(1000)  # Generate a sample of 1000 random numbers
beta <- 0.5          # Subset size is n^beta
s <- 100             # Number of subsets
r <- 500             # Resampling iterations per subset
```

```r
u <- mean             # Point estimate function (mean)
xi <- function(resamples, estimate) {
  quantile(resamples, c(0.025, 0.975))  # 95% confidence interval
}

result <- BLB(data, beta, s, r, u, xi)
print(result)
```

Example Usage:

```
##        2.5%       97.5%
## -0.05322934  0.06436710
```

2. `ScaSub`

The `ScaSub` function implements a subsampling-based method for scalable statistical analysis.

Parameters:

- `data`: A numeric vector of data values.
- `beta`: A numeric value determining the size of each subsample block, `b = n^beta`.
- `c1`: The stride length as a fraction of the block size.
- `u`: A function to calculate the point estimate (e.g., `mean`, `median`).
- `xi`: A function to compute a confidence interval or other summary from the resampling results.

```r

data <- rnorm(1000)  # Generate a dataset of 1000 random numbers
beta <- 0.5          # Subsample block size is n^beta
c1 <- 0.2            # Stride length
u <- mean            # Point estimate function (mean)
xi <- function(root_dist, estimate) {
  quantile(root_dist, c(0.025, 0.975))  # 95% confidence interval
}

result <- ScaSub(data, beta, c1, u, xi)
print(result)
```

Example Usage:

```
##        2.5%       97.5%
## -0.04492937  0.06353930
```

3. `BLBpar`

The `BLBpar` function is a parallelized implementation of the Bag of Little Bootstraps algorithm, which leverages multiple CPU cores for faster computation.

Parameters:

- `data`: A numeric vector of data values.
- `beta`: A numeric value determining the size of each subset, `b = n^beta`.
- `s`: The number of subsets to generate.
- `r`: The number of resampling iterations for each subset.
- `u`: A function to calculate the point estimate (e.g., `mean`, `median`).
- `xi`: A function to compute a confidence interval or other summary from the resampling results.

```r
data <- rnorm(1000)  # Generate a sample of 1000 random numbers
beta <- 0.5          # Subset size is n^beta
s <- 100             # Number of subsets
r <- 500             # Resampling iterations per subset
u <- mean            # Point estimate function (mean)
xi <- function(resamples, estimate) {
  quantile(resamples, c(0.025, 0.975))  # 95% confidence interval
}

result <- BLBpar(data, beta, s, r, u, xi)
print(result)
```

Example Usage:

```
##        2.5%       97.5%
## -0.06278178  0.04862072
```

4. Comparing the Running Speed of `BLB` and `BLBpar`

One of the key advantages of the BLBpar function is its ability to significantly speed up the computation by utilizing multiple CPU cores for parallel processing. In contrast, the BLB function runs on a single core, which can be slower when dealing with large datasets or a large number of resampling iterations.

We will compare the running time of the `BLB` and `BLBpar` functions using the same dataset and parameters.

```r
# Generate a sample of 1000 random numbers
data <- rnorm(10000)
beta <- 0.5
s <- 100
r <- 500
u <- mean
xi <- function(resamples, estimate) {
  quantile(resamples, c(0.025, 0.975))  # 95% confidence interval
}

# Time for BLB function
start_time <- proc.time()
BLB(data, beta, s, r, u, xi)
```

```
##         2.5%        97.5%
## -0.030055880  0.008949334
```

```r
blb_time <- proc.time() - start_time

# Time for BLBpar function
start_time <- proc.time()
BLBpar(data, beta, s, r, u, xi)
```

```
##        2.5%       97.5%
## -0.01615728  0.02240615
```

```r
blbpar_time <- proc.time() - start_time

# Print the results
print(paste("BLB running time: ", round(blb_time["elapsed"], 2), "seconds"))
```

```
## [1] "BLB running time:  33.54 seconds"
```

```
print(paste("BLBpar running time: ", round(blbpar_time["elapsed"], 2), "seconds"))
```

```
## [1] "BLBpar running time:  7.27 seconds"
```

Explanation

- The proc.time() function is used to measure the elapsed time for both the BLB and BLBpar functions.
- By comparing the running time, we can observe how much faster BLBpar is compared to BLB, especially when the number of subsets (s) and resampling iterations (r) is large.

Conclusion

The `BLBtools6520` package provides scalable and efficient methods for bootstrap and subsampling analyses, suitable for large datasets. By using the examples provided above, you can incorporate these methods into your own data analysis workflows. For additional help, refer to the package documentation.