Bag of Little Bootstrap

Jaymie Tam

Introduction

This project is to create, modify and improve the bag of little bootstrap R packages. The R package is called the blblm and it is designed primarily for use by individuals who are interested in data analysis to help them perform a weighted bootstrap linear regression. Additionally, this package provides a list of other functions to estimates based on the linear regression. This package is useful for data which may not meet the assumption of parametric linear regression such as small sample sizes, asymmetric statistical distribution. Since some students may only want to have preliminary understanding of the data, this r package may help them to achieve their result.

**The restriction of the function does not omit na datas, to further usage of the package, please use na.omit(data) before using the function.

```
library(blblm)
library(parallel)
library(bench)
df <- data.frame(mtcars)
df <- na.omit(mtcars)
```

Features

Bootstrap Linear Regression(blblm)

Based on the user's interest in data, the function performs bootstrap linear regression by splitting data into subsamples and sample n from b with replacement which is different from ordinary bootstrap.

Coefficient of the desired explanatory/independent variable

```
#arr_delay time against Sepal
#m = 3 subsamples
#B = 100 bootstraps
model <- blblm(mpg ~ wt * hp, data = df, m = 3, B = 1000)
coef(model)
#> (Intercept)          wt          hp       wt:hp
#> 54.23771052 -9.36041234 -0.12961018  0.02980846
```

-Formula of the regression model

```
formula(model)
#> mpg ~ wt * hp
```

Estimates of each bootstrap coefficient result

```
#first and second output of the estimates from bootstrap
model$estimates$`1`[[1]]
#> $coef
#>  (Intercept)          wt          hp       wt:hp
#>  96.87483371 -22.09425229  -0.34559426   0.09401864
#>
#> $sigma
#> [1] 1.919568
model$estimates$`1`[[2]]
#> $coef
#>  (Intercept)          wt          hp       wt:hp
#>  59.53367345 -10.74478572  -0.14675752   0.03576552
#>
#> $sigma
#> [1] 2.009695
```

Bootstrap Linear Regression with Parallelization
It performs the same result as the Bootstrap Linear Regression(blblm), and it adds a feature of parallelization to speed up for larger dataset.

```
cl <- makeCluster(2)
#cl = cluster user uses
model2 <- blblm_par(mpg ~ wt * hp, data = mtcars, m = 3, B = 100, cl)
parallel::stopCluster(cl)
```

Formula of the regression model

```
model2$formula
#> mpg ~ wt * hp
```

Coefficient of the desired explanatory/independent variable

```
coef(model2)
#> (Intercept)          wt          hp       wt:hp
#> 53.90038348 -9.49956494 -0.15805925  0.03830444
```

Estimates of each bootstrap coefficient result

```
#first output of the estimates from bootstrap
model2$estimates$`1`[[1]]
#> $coef
#>  (Intercept)          wt          hp       wt:hp
#>  54.04082104 -10.01092798  -0.16064240   0.04505663
#>
#> $sigma
#> [1] 1.940371
model2$estimates$`1`[[2]]
#> $coef
```

```
#>  (Intercept)           wt           hp        wt:hp
#>  56.72063775 -11.89324660  -0.16432618   0.05054535
#>
#> $sigma
#> [1] 1.719332
```

Confidence Interval(confint)
Confidence Interval for explanatory variables in a user's desired confidence level.

```
confint(model, level = 0.95)
#>                2.5%       97.5%
#> wt    -13.888717643 -4.93398096
#> hp     -0.212970933 -0.04757089
#> wt:hp   0.005854255  0.05436646
```

Predict New Data
It allows user to predict new data based on the linear regression model

```
pred <- predict(model, data.frame(wt = c(rnorm(100)), hp = c(rnorm(100))))
head(pred)
#>        1        2        3        4        5        6
#> 54.19042 37.06338 36.78893 50.76855 55.77607 55.75881
#user can choose to include condifence level
pred2 <- predict(model, data.frame(wt = c(rnorm(100)), hp = c(rnorm(100))),
  confidence = TRUE)
head(pred2)
#>        fit      lwr      upr
#> 1 59.63149 42.44103 76.44136
#> 2 65.17596 45.59221 84.62403
#> 3 53.99876 39.47798 68.10677
#> 4 52.49303 38.72991 65.87695
#> 5 42.99194 33.86302 51.83207
#> 6 48.24699 36.68708 59.59133
```

Sigma with confidence level Compute the confidence interval for sigma

```
 sigma(model, confidence = TRUE)
#>    sigma      lwr      upr
#> 1.597436 1.170464 1.935934
```

Source of Package

Rcpp

The use of C++ library RcppArmadillo allows us to compute various matrix multiplication and decomposition to improve the speed of the function. This package include a blblm or blblm_par function which is in the use of fastlm function. In the use of linear algebra the function was able to compute coef, weight residual and sigma by matrix transformation and multiplication.

Rcpp fastlm function computes of the following estimates:
coefficient, residuals, rank, degree of freedom, sigma

```
//Rcpp function for weight linear regression
#include <RcppArmadillo.h>
// [[Rcpp::depends(RcppArmadillo)]]

using namespace Rcpp;

// [[Rcpp::export]]
List fastlm(const arma::mat& X, const arma::colvec& y, const arma::vec& wi) {
    int n = X.n_rows, k = X.n_cols, r = rank(X);

    arma::mat weight = diagmat(wi); //get the weight vector

    arma::colvec coef = arma::solve(((trans(X))*weight*X) , ((trans(X)*weight)*y));
    // fit model y ~ X

    arma::colvec res  = y - X*coef;           // residuals

    double weight_sum = accu(wi);

    double num = arma::as_scalar(( accu( weight * (pow(res,2.0)) )));
    double denom = weight_sum - r;
    double s =  sqrt(num / denom);

    return List::create(Named("coefficients") = coef,
                        Named("weight sum")   = weight_sum,
                        Named("sigma")         = s,
                        Named("df.residual")  = n - k,
                        Named("rank")         = r,
                        Named("response")     = y,
                        Named("Res")  = res,
                        Named("weight") = wi);
}
```

Compare the time of fastlm with weighted linear regression(lm.wif) and it is two times faster than the speed of weighted lm.

```
#transform the response and explanatory variable to fit the function
x <- model.matrix(mpg ~ wt * hp , mtcars)
y <- model.response(model.frame(mpg ~ wt * hp, mtcars))
w <- rmultinom(1, 10, rep(1, nrow(mtcars)))

#comapre three lm functions
bench::mark(
  fastlm = {fastlm(x,y,w)},
  weighted_lm = {lm.wfit(x,y,w)},
  check = FALSE, relative = TRUE
)
#> # A tibble: 2 x 6
```

```
#>   expression    min median `itr/sec` mem_alloc `gc/sec`
#>   <bch:expr>  <dbl>  <dbl>     <dbl>     <dbl>    <dbl>
#> 1 fastlm       1      1         2.02       1       1
#> 2 weighted_lm  2.22   2.16      1         29.4     4.96
```

Parallelization

The use of library(parallel) allows us to create the blblm_par function to include parallelizatoin for user to choose to work with cluster to speed up the process.

The following demostrated the speed of using 2 CPU is almost two times faster than the orignial blblm. It may benefit users who want to process large datasets.

```
cl <- makeCluster(2)
bench::mark(
  blblm_par(mpg ~ wt * hp, data = mtcars, m = 2, B = 1000, cl),
  blblm(mpg ~ wt * hp, data = mtcars, m = 2, B = 1000),
  check = FALSE, relative = TRUE
)
#> # A tibble: 2 x 6
#>   expression                                                  min median
#>   <bch:expr>                                                <dbl>  <dbl>
#> 1 blblm_par(mpg ~ wt * hp, data = mtcars, m = 2, B = 1000, cl)  1      1
#> 2 blblm(mpg ~ wt * hp, data = mtcars, m = 2, B = 1000)          1.77   1.77
#> # … with 3 more variables: `itr/sec` <dbl>, mem_alloc <dbl>, `gc/sec` <dbl>

stopCluster(cl)
```

Cited: Armadillo Library http://arma.sourceforge.net/