

# 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.

```
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
#>  47.1826100  -7.2390188  -0.1048620   0.0224022
```

-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
#> 52.88780664 -9.93628410 -0.12955557  0.03503494
#>
#> $sigma
#> [1] 2.55343
model$estimates$`1`[[2]]
#> $coef
#> (Intercept)          wt          hp          wt:hp
#> 51.83591432 -8.75188314 -0.12003131  0.02856916
#>
#> $sigma
#> [1] 2.326753
```

### 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
#> 47.09474811 -7.26548226 -0.10591331  0.02330666
```

Estimates of each bootstrap coefficient result

```
#first output of the estimates from bootstrap
model2$estimates$`1`[[1]]
#> $coef
#> (Intercept)          wt          hp          wt:hp
#> 41.52559135 -5.81564588 -0.05811072  0.01087011
#>
#> $sigma
#> [1] 0.8778837
model2$estimates$`1`[[2]]
#> $coef
#> (Intercept)          wt          hp          wt:hp
#> 40.232686831 -4.891519719 -0.049496768  0.006250417
#>
#> $sigma
#> [1] 1.335962
```

### 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      -10.402954941 -4.05055953
#> hp       -0.182886496 -0.04964249
#> wt:hp    0.006049382  0.03924932
```

### 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
#> 51.30198 47.89488 35.50585 52.18316 39.91020 49.63635
#user can choose to include confidence level
pred2 <- predict(model, data.frame(wt = c(rnorm(100)), hp = c(rnorm(100))),
  confidence = TRUE)
head(pred2)
#>           fit          lwr          upr
#> 1 61.90480 46.43026 77.68497
#> 2 44.72844 36.45968 53.02302
#> 3 30.97063 27.69869 34.89791
#> 4 54.96695 42.57561 67.59018
#> 5 51.72331 40.59613 62.93817
#> 6 31.11563 27.82209 35.02020
```

## Sigma with confidence level

Compute the confidence interval for sigma

```
sigma(model, confidence = TRUE)
#>      sigma      lwr      upr
#> 1.880122 1.300056 2.305645
```

## 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.wfit) 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       1.86         1       1
#> 2 weighted_lm  2.11    1.89       1       29.4     4.86

```

## 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 demonstrated the speed of using 2 CPU is almost two times faster than the original 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)      1.72    1.72  
#> # ... with 3 more variables: `itr/sec` <dbl>, mem_alloc <dbl>, `gc/sec` <dbl>  
stopCluster(cl)
```

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