STA 141C Final Project - blblm2

```
library(blblm2)
library(bench)
library(future)
library(tidyverse)
library(ggplot2)
```

The Original blblm Package

The original blblm package used bag of little bootstraps to estimate a linear regression model. The blblm() function estimated the model, and users could use coef(), sigma(), confint(), and predict() to see the coefficients, standard deviation, confidence interal, and pedictions using new data, respectfully. The blblm2 package performs the same; however, several improvements and changes have been made in the process.

Improvements/changes include adding a parallelization option to blblm() to speed up computations on large data sets, writing a new function called blblog() which performs bag of little bootstraps on a logistic regression model, and writing a fast version of blblm() using Rcpp.

Examples of how blblm performs without parallelization

First, I shall generate a data set to use for the following examples.

```
df <- data.frame(y = rnorm(n = 1000, mean = 100, sd = 10),
                 X = rnorm(n = 1000, mean = 100, sd = 10))
```

Create the bag of little bootstrapped linear model using blblm() with subsamples and 100 repeats of the bootstrap procedure.

```
fit <- blblm(y ~ X, data = df, m = 3, B = 100)
```

fit is a blblm object.

```
class(fit)
#> [1] "blblm"
```

The function print() prints the blblm model

```
print(fit)
#> blblm model: y ~ X
```

The function coef() computes the coefficients of the linear regression model. The function must work because the estimate intercept is close to the mean specified when creating the data frame.

```
coef(fit)
#> (Intercept)           X
```

```
#> 98.13836855  0.02004905
```

The function sigma() computes the standard deviation of the blblm model. Note, by setting confidence = TRUE, one can compute a confidence interval for sigma and specify the level. The example below shows a 90% confidence interval for sigma.

```
sigma(fit)
#> [1] 10.03304
sigma(fit, confidence = TRUE, level = .90)
#>     sigma       lwr       upr
#> 10.033042  9.580613 10.482836
```

One can also compute a confidence interval for the coefficient estimates.The example below shoes the 90% confidence interval for the variable X.

```
confint(fit, level = 0.90)
#>            5%        95%
#> X -0.03300568 0.07547623
```

Furthermore, by using the function predict() one can generate predictions on new data based on a fitted blblm model. To demonstrate this I will generate new data.

```
new_data <- data.frame(y = rnorm(n = 20, mean = 100, sd = 10),
                 X = rnorm(n = 20, mean = 100, sd = 10))
```

```
predict(fit, new_data, confidence = TRUE) %>% head(1)
#>        fit      lwr      upr
#> 1 99.99166 99.2684 100.7296
```

Improvements

Parallelization of blblm()

To improve performance of the blblm function, I added multi-core processing to the function. The argument parallel lets users specify if they would like to use parallelization when using blblm(). The parallel argument is default to FALSE. Users must specify TRUE if they would like to use parallelization. Parallelization is done using future_map() the package, furrr. In addition to specifying parallel = TRUE, users must also use future::plan() to specify the number of cores wished to be used in the computation.

The benchmark below shows blblm() with no parallelization vs blblm() with parallelization using four cores. Please note that performance is only increased when using large data sets. I generated a large data set below as an example for the benchmark.

```
big_data <- data.frame(y = rnorm(n = 10000, mean = 100, sd = 10),
                 a = rnorm(n = 10000, mean = 100, sd = 10),
                 B = rnorm(n = 10000, mean = 100, sd = 10),
                 C = rnorm(n = 10000, mean = 100, sd = 10),
                 D = rnorm(n = 10000, mean = 100, sd = 10),
                 E = rnorm(n = 10000, mean = 100, sd = 10))
```

I will use four cores when doing paralellization.

```
plan(multiprocess, workers = 4)
```

```
bm <- mark(blblm(a ~ B + C + D + E, data = big_data, m = 3, B = 1000),
     blblm(a ~ B + C + D + E, data = big_data, m = 3, B = 1000, parallel = TRUE), check = FALSE)
```

Performance improved twice fold when using parallelization. Using the argument parallel = TRUE will save a lot of time on large data sets.

New functions

blblog Logistic Regression

A new addition I added to the package was the function blblog() which uses bag of little bootstraps to estimate a logistic regression model. Logistic regression was implemented by using glm.fit() with the family = binomial().

I will use the iris data set for the following example. The example below creates the bootstrapped logistic regression model using two subsizes and 100 repeats.

```
log <- blblog(Species ~ Sepal.Length, data = iris, m = 2, B = 100)
```

```
print(log)
#> blblog model: Species ~ Sepal.Length
```

The blblog model is of class “blblog”

```
class(log)
#> [1] "blblog"
```

The estimate coefficients are as follows.

```
coef(log)
#>  (Intercept) Sepal.Length
#>    -30.31295      5.63332
```