Package demo

William Shih, Ricardo Simpao, Nilay Varshney, Luke Yee

3/20/2020

Here is an example of how our package functions run. For our data set, we are using a "SGEMM GPU kernel performance Data Set," which measures the running times of a matrix-matrix product, given different parameter combinations.

```
library(devtools)
library(tidyverse)
#library(STA141CFinal)
library(Rcpp)
library(RcppArmadillo)
```

```
## Warning: package 'RcppArmadillo' was built under R version 3.6.3
```

```
library(tidyverse)
library(furrr)

sourceCpp("src/coef_CI_C.cpp")
```

```
## Warning in normalizePath(path.expand(path), winslash, mustWork): path[1]="C:/
## Users/williamshih/Google Drive (wshih@ucdavis.edu)/2019-2020 Q2/STA 141C/
## STA141CFinal/src/../inst/include": The system cannot find the path specified
```

```
sourceCpp("src/linear_reg_bs_C.cpp")
```

```
## Warning in normalizePath(path.expand(path), winslash, mustWork): path[1]="C:/
## Users/williamshih/Google Drive (wshih@ucdavis.edu)/2019-2020 Q2/STA 141C/
## STA141CFinal/src/../inst/include": The system cannot find the path specified
```

```
sourceCpp("src/PI_C.cpp")
```

```
## Warning in normalizePath(path.expand(path), winslash, mustWork): path[1]="C:/
## Users/williamshih/Google Drive (wshih@ucdavis.edu)/2019-2020 Q2/STA 141C/
## STA141CFinal/src/../inst/include": The system cannot find the path specified
```

```
sourceCpp("src/s2_CI_C.cpp")
```

```
## Warning in normalizePath(path.expand(path), winslash, mustWork): path[1]="C:/
## Users/williamshih/Google Drive (wshih@ucdavis.edu)/2019-2020 Q2/STA 141C/
## STA141CFinal/src/../inst/include": The system cannot find the path specified
```

```
files = list.files(path = "R/")
files = paste0("R/", files)[c(-7,-10)]
sapply(files, source)
```

```
##         R/coef_CI.R R/coef_CI_par.R R/linear_reg_bs.R R/linear_reg_bs_par.R
## value   ?           ?               ?                 ?
## visible FALSE       FALSE           FALSE             FALSE
##         R/PI.R R/PI_par.R R/s2_CI.R R/s2_CI_par.R
```

```
## value   ?      ?          ?         ?
## visible FALSE  FALSE      FALSE     FALSE
```

```
dat = read_csv("sgemm_product.csv")
dat = dat[1:20000,]
```

```
#We specifiy a specific column set
y = dat$`Run1 (ms)`
x = dat[,1:(ncol(dat)-4)]

#linear model objects
#fit = linear_reg_bs(x = x, y = y, s = 10, r = 1000)
#fit$bootstrap_coefficient_estimates

#fit2 = linear_reg_bs_par(x = x, y = y, s = 10, r = 1000)
#fit2$bootstrap_coefficient_estimates

fit = linear_reg_bs_C(x, y, s = 10, r = 1000)
```

Linear Regression with blb

95 % Confidence Interval for Variable Coefficients

```
coef_CI(fit, alpha = 0.05)
```

```
##            Lower_Bounds     Estimates Upper_Bounds
## Intercept  70.424863865  73.325552172  76.12228266
## MWG         0.101003795   0.192702339   0.28548683
## NWG        -0.002710242   0.008315449   0.01950888
## KWG         0.709980041   0.757410640   0.80425252
## MDIMC       1.078983758   1.183896307   1.28753219
## NDIMC      -0.242901595  -0.191425866  -0.14102122
## MDIMA      -0.131571137  -0.041932712   0.04799987
## NDIMB      -0.485223506  -0.446726495  -0.40792001
## KWI        -0.241655391  -0.130694078  -0.01886823
## VWM        -8.960579097  -8.197622623  -7.43060286
## VWN        -4.801640957  -4.570059217  -4.33830712
## STRM       -0.084279422   0.590364566   1.26826343
## STRN       -5.443384241  -4.786309959  -4.11725447
## SA        -16.695588000 -16.025977906 -15.35426089
## SB          2.146966956   2.811664053   3.47800830
```

```
coef_CI_par(fit,alpha = 0.05)
```

```
##            Lower_Bounds     Estimates Upper_Bounds
## Intercept  70.424863865  73.325552172  76.12228266
## MWG         0.101003795   0.192702339   0.28548683
## NWG        -0.002710242   0.008315449   0.01950888
## KWG         0.709980041   0.757410640   0.80425252
## MDIMC       1.078983758   1.183896307   1.28753219
## NDIMC      -0.242901595  -0.191425866  -0.14102122
## MDIMA      -0.131571137  -0.041932712   0.04799987
## NDIMB      -0.485223506  -0.446726495  -0.40792001
## KWI        -0.241655391  -0.130694078  -0.01886823
```

```
## VWM        -8.960579097  -8.197622623  -7.43060286
## VWN        -4.801640957  -4.570059217  -4.33830712
## STRM       -0.084279422   0.590364566   1.26826343
## STRN       -5.443384241  -4.786309959  -4.11725447
## SA        -16.695588000 -16.025977906 -15.35426089
## SB          2.146966956   2.811664053   3.47800830
```

```
coef_CI_C(fit,alpha = 0.05)
```

```
##            Lower_Bounds     Estimates Upper_Bounds
## Intercept  70.425477472  73.325552172  76.13883460
## MWG         0.101013321   0.192702339   0.28630050
## NWG        -0.002708709   0.008315449   0.01962176
## KWG         0.709988445   0.757410640   0.80463206
## MDIMC       1.078997406   1.183896307   1.28844520
## NDIMC      -0.242890114  -0.191425866  -0.14063939
## MDIMA      -0.131550294  -0.041932712   0.04881628
## NDIMB      -0.485212997  -0.446726495  -0.40751785
## KWI        -0.241635361  -0.130694078  -0.01825898
## VWM        -8.960464282  -8.197622623  -7.42635395
## VWN        -4.801572531  -4.570059217  -4.33616023
## STRM       -0.084092810   0.590364566   1.27594105
## STRN       -5.443301161  -4.786309959  -4.11053601
## SA        -16.695373264 -16.025977906 -15.34868521
## SB          2.147086665   2.811664053   3.48255620
```

```
(b1 = bench::mark(
  coef_CI(fit, alpha = 0.05),
  coef_CI_par(fit,alpha = 0.05),
  coef_CI_C(fit, alpha = 0.05),
  check = FALSE)
)
```

```
## # A tibble: 3 x 6
##   expression                          min   median `itr/sec` mem_alloc `gc/sec`
##   <bch:expr>                        <bch:tm> <bch:tm>     <dbl> <bch:byt>    <dbl>
## 1 coef_CI(fit, alpha = 0.05)       24.38ms     30ms      32.8    7.49MB     7.57
## 2 coef_CI_par(fit, alpha = 0.05)   47.93ms   49.1ms      20.3     7.8MB     8.72
## 3 coef_CI_C(fit, alpha = 0.05)      6.98ms    7.3ms     136.     1.15MB     4.19
```

Notice that `coef_CI_par` offers better memory allocation than `coef_CI`.

```
plan(multiprocess, workers = 4)
PI(fit, dat[1:3, 1:14], alpha = 0.05)
PI_par(fit, dat[1:3, 1:14], alpha = 0.05)
PI_C(fit, data[1:3, 1:14], alpha = 0.05)

(b2 = bench::mark(
  PI(fit, x, alpha = 0.05),
  PI_par(fit, x, alpha = 0.05),
  PI_C(fit, x, alpha = 0.05),
  check = FALSE
)
```

95 % Confindence Interval for Variance

```
s2_CI(fit, alpha = 0.05)
```

```
## Lower_Bound    Estimate Upper_Bound
##    5630.268    5832.922    6042.915
```

```
s2_CI_par(fit, alpha = 0.05)
```

```
## Lower_Bound    Estimate Upper_Bound
##    5630.268    5832.922    6042.915
```

```
s2_CI_C(fit, alpha = 0.05)
```

```
## Lower_Bound    Estimate Upper_Bound
##    5630.317    5832.922    6044.210
```

```
(b3 = bench::mark(
  s2_CI(fit, alpha = 0.05),
  s2_CI_par(fit, alpha = 0.05),
  s2_CI_C(fit, alpha = 0.05),
  check = FALSE)
)
```

```
## # A tibble: 3 x 6
##   expression                      min   median `itr/sec` mem_alloc `gc/sec`
##   <bch:expr>                  <bch:tm> <bch:tm>     <dbl> <bch:byt>    <dbl>
## 1 s2_CI(fit, alpha = 0.05)      1.65ms   1.81ms     525.    118.1KB     6.21
## 2 s2_CI_par(fit, alpha = 0.05) 22.07ms  23.52ms      42.8   413.3KB    10.1
## 3 s2_CI_C(fit, alpha = 0.05)    89.5us   91.1us    9386.      7.1KB     0
```

Notice that `s2_CI_par` offers better memory allocation than `s2_CI`.