TestingRandC

William Shih

3/17/2020

```
library(Rcpp)
library(RcppArmadillo)
library(rbenchmark)
set.seed(121)
sourceCpp("C/calc_slope.cpp")
sourceCpp("C/linreg.cpp")
sourceCpp("C/lr_coefficient_CI.cpp")
source("R/calc_slope.R")
source("R/linreg.R")
source("R/lr_coefficient_CI.R")
```

Benchmark calculating slope of p = 1, n = 200,000 with 1000 replications

test	replications	elapsed	relative	user.self	sys.self
C++	1000	1.48	1.00	1.06	0.40
R	1000	4.78	3.23	3.48	1.29

Benchmark p = 400 and n = 200,000 Linear Regression with 100 replications

	test	replications	elapsed	relative	user.self	sys.self
2	.lm.fit	100	5.65	2.100	4.20	1.03
6	C++ with RcppArmadillo	100	2.69	1.000	1.54	1.12
5	C++ with std::inner_product	100	14.38	5.346	13.56	0.73
4	C++ without std::inner_product	100	4.91	1.825	4.27	0.63
1	lm	100	11.36	4.223	8.67	1.92
3	lm.fit	100	4.47	1.662	3.40	1.06
7	R	100	3.95	1.468	2.85	1.09

Benchmark p = 1 and n = 200,000 Linear Regression with 100 replications

	test	replications	elapsed	relative	user.self	sys.self
2	.lm.fit	100	2.16	1.421	1.66	0.39
6	C++ with RcppArmadillo	100	1.52	1.000	0.77	0.74
5	C++ with std::inner_product	100	5.02	3.303	4.47	0.51
4	C++ without std::inner_product	100	1.89	1.243	1.39	0.38
1	lm	100	7.01	4.612	6.12	0.70
3	lm.fit	100	2.35	1.546	1.80	0.46

	test	replications	elapsed	relative	user.self	sys.self
7	R	100	2.05	1.349	1.43	0.57

Benchmark p = 40 and n = 200,000 Linear Regression with 1 replication

	test	replications	elapsed	relative	user.self	sys.self
2	.lm.fit	1	0.64	1.778	0.63	0.02
6	C++ with RcppArmadillo	1	0.36	1.000	0.27	0.09
5	C++ with std::inner_product	1	5.26	14.611	5.22	0.03
4	C++ without std::inner_product	1	5.58	15.500	5.51	0.03
1	lm	1	0.75	2.083	0.67	0.08
3	lm.fit	1	0.64	1.778	0.57	0.08
7	R	1	0.54	1.500	0.50	0.05

Benchmark multiplying 40*200000 by 200000*40 matrix

	test	replications	elapsed	relative	user.self	sys.self
1	%*%	1	0.31	1.000	0.29	0.01
4	C++ with RcppArmadillo	1	0.34	1.097	0.26	0.08
3	C++ with std::inner_product	1	2.48	8.000	2.47	0.00
2	C++ without std::inner_product	1	4.45	14.355	4.45	0.00

Benchmark multiplying 4*200000 by 200000*4 matrix

	test	replications	elapsed	relative	user.self	sys.self
1	%*%	100	0.95	1.000	0.83	0.11
4	C++ with RcppArmadillo	100	1.69	1.779	1.05	0.62
3	C++ with std::inner_product	100	3.08	3.242	2.83	0.25
2	C++ without std::inner_product	100	1.46	1.537	1.22	0.22

Benchmark calling t distribution with 100,000 replications

	test	replications	elapsed	relative	user.self	sys.self
2	C++ calling R	1e+05	3.36	5.508	3.34	0.00
1	C++ using Boost	1e+05	0.80	1.311	0.78	0.02
3	R	1e+05	0.61	1.000	0.61	0.00

Benchmark confidence interval of linear regression result with 10,000 replications

test	replications	elapsed	relative	user.self	sys.self
C++	10000	0.15	1.000	0.12	0.01
R	10000	0.41	2.733	0.40	0.00

Code Appendix

```
knitr::opts_chunk$set(echo = TRUE)
library(Rcpp)
library(RcppArmadillo)
library(rbenchmark)
set.seed(121)
sourceCpp("C/calc_slope.cpp")
sourceCpp("C/linreg.cpp")
sourceCpp("C/lr_coefficient_CI.cpp")
source("R/calc_slope.R")
source("R/linreg.R")
source("R/lr_coefficient_CI.R")

Random = runif(200000,0,100)
AT = data.frame(x = 1:200000, y = 1:200000 + Random)


x1 = runif(200000,0,1000) + runif(200000, 500, 1000)
x2 = runif(200000,0,2000) + runif(200000, 1000, 2000)
x3 = runif(200000,2000,3000) + runif(200000, 1000, 2000)
x4 = runif(200000,2500,2750) + runif(200000, 1000, 2000)
y = x1 + x2 + x3 + x4 + Random^2 + runif(20000, 500, 777)
xFrame = as.matrix(data.frame(x1,x2,x3,x4))




knitr::kable((benchmark("C++" = {calc_slopeC(AT)},
          "R" = {calc_slope(AT)},
          replications = 1000,
          columns = c("test","replications","elapsed","relative","user.self","sys.self"))))
#p = 4
knitr::kable((benchmark("lm" = {lm(y ~ xFrame)},
          ".lm.fit" = {.lm.fit(as.matrix(data.frame(1, xFrame)), y)},
          "lm.fit" = {lm.fit(as.matrix(data.frame(1, xFrame)), y)},
          "C++ without std::inner_product" = {linear_regC(xFrame, y)},
          "C++ with std::inner_product" = {linear_regC2(xFrame, y)},
          "C++ with RcppArmadillo" = {linear_regC3(xFrame, y)},
          "R" = {linear_reg(xFrame, y)},
            replications = 100,
            columns = c("test","replications","elapsed","relative","user.self","sys.self"))))
# p =1
knitr::kable((benchmark("lm" = {lm(AT[[2]] ~ AT[[1]])},
          ".lm.fit" = {.lm.fit(as.matrix(data.frame(1, AT[[1]])), as.vector(AT[[2]]))},
          "lm.fit" = {lm.fit(as.matrix(data.frame(1, AT[[1]])), as.vector(AT[[2]]))},
          "C++ without std::inner_product" = {linear_regC(as.matrix(AT[[1]]), as.vector(AT[[2]]))},
          "C++ with std::inner_product" = {linear_regC2(as.matrix(AT[[1]]), as.vector(AT[[2]]))},
          "C++ with RcppArmadillo" = {linear_regC3(as.matrix(AT[[1]]), as.vector(AT[[2]]))},
          "R" = {linear_reg(as.matrix(AT[[1]]), as.vector(AT[[2]]))},
          replications = 100,
          columns = c("test","replications","elapsed","relative","user.self","sys.self"))))
#p = 40
```

```
x1 = runif(8000000,0,1000) + runif(8000000, 500, 1000)
FourtyX = matrix(x1, nrow = 200000, ncol = 40)
knitr::kable((benchmark("lm" = {lm(y ~ FourtyX)},
          ".lm.fit" = {.lm.fit(as.matrix(data.frame(1, FourtyX)), y)},
          "lm.fit" = {lm.fit(as.matrix(data.frame(1, FourtyX)), y)},
          "C++ without std::inner_product" = {linear_regC(FourtyX, y)},
          "C++ with std::inner_product" = {linear_regC2(FourtyX, y)},
          "C++ with RcppArmadillo" = {linear_regC3(FourtyX, y)},
          "R" = {linear_reg(FourtyX, y)},
          replications = 1,
          columns = c("test","replications","elapsed","relative","user.self","sys.self"))))
knitr::kable(benchmark("%*%" = {t(FourtyX) %*% FourtyX},
          "C++ without std::inner_product" = {multiply(t(FourtyX), FourtyX)},
          "C++ with std::inner_product" = {multiply2(t(FourtyX), FourtyX)},
          "C++ with RcppArmadillo" = {armamultiply(t(FourtyX), FourtyX)},
          replications = 1,
          columns = c("test","replications","elapsed","relative","user.self","sys.self")))
knitr::kable(benchmark("%*%" = {t(xFrame) %*% xFrame},
          "C++ without std::inner_product" = {multiply(t(xFrame), xFrame)},
          "C++ with std::inner_product" = {multiply2(t(xFrame), xFrame)},
          "C++ with RcppArmadillo" = {armamultiply(t(xFrame), xFrame)},
          replications = 100,
          columns = c("test","replications","elapsed","relative","user.self","sys.self")))

knitr::kable(benchmark("C++ using Boost" = {tc(0.99, 55)},
          "C++ calling R" = {tr(0.99,55)},
          "R" = {qt(0.99, 55)},
          replications = 100000,
          columns = c("test","replications","elapsed","relative","user.self","sys.self")))

z = linear_reg(FourtyX, y)
knitr::kable(benchmark("C++" = {lr_coefficient_CI_C(z, 0.95)},
          "R" = {lr_coefficient_CI(z, 0.95)},
          replications = 10000,
          columns = c("test","replications","elapsed","relative","user.self","sys.self")))
```