fRLR package: Fit Repeated Linear Regressions

Lijun Wang

January 8, 2019

Contents

1 Introduction

This R package aims to fit *Repeated Linear Regressions* in which there are some same terms.

2 An Example

Let's start with the simplest situation, we want to fit a set of regressions which only differ in one variable. Specifically, denote the response variable as y, and these regressions are as follows.

$$\begin{array}{ll} y & \sim x_1 + cov_1 + cov_2 + \ldots + cov_m \\ y & \sim x_2 + cov_1 + cov_2 + \ldots + cov_m \\ \cdot & \sim \cdots \\ y & \sim x_n + cov_1 + cov_2 + \ldots + cov_m \end{array}$$

where $cov_i, i = 1, \ldots, m$ are the same variables among these regressions.

3 Ideas

Intuitively, we can finish this task by using a simple loop.

However, it is not efficient in that situation. As we all know, in the linear regression, the main goal is to estimate the parameter β. And we have

$$\hat{\beta} = (X'X)^{-1}X'Y$$

where X is the design matrix and Y is the observation of response variable.

It is obvious that there are some same elements in the design matrix, and the larger m is, the more elements are the same. So I want to reduce the cost of computation by separating the same part in the design matrix.

4 Method

For the above example, the design matrix can be denoted as $X = (x, cov)$. If we consider intercept, it also can be seen as the same variable among these regression, so it can be included in cov naturally. Then we have

$$(X'X)^{-1} = \begin{bmatrix} x'x & x'cov \\ cov'x & cov'cov \end{bmatrix} = \begin{bmatrix} a & v' \\ v & B \end{bmatrix}$$

Woodbury formula tells us

$$(A + UCV)^{-1} = A^{-1} - A^{-1}U(C^{-1} + VA^{-1}U)^{-1}VA^{-1}$$

Let

$$A = \begin{bmatrix} a & O \\ O & B \end{bmatrix}, U = \begin{bmatrix} 1 & 0 \\ O & v \end{bmatrix}, V = \begin{bmatrix} 0 & v' \\ 1 & O \end{bmatrix}$$

and $C = I_{2\times 2}$. Then we can apply woodbury formula,

$$(X'X)^{-1} = (A + UCV)^{-1} = A^{-1} - A^{-1}U(C^{-1} + VA^{-1}U)^{-1}VA^{-1}$$

where

$$A^{-1} = \begin{bmatrix} a^{-1} & O \\ O & B^{-1} \end{bmatrix}$$

We can do further calculations to simplify and obtain the following result

$$(X'X)^{-1} = \begin{bmatrix} 1/a + \frac{a}{a-v'B^{-1}v} v'B^{-1}v & -\frac{v'B^{-1}}{a-v'B^{-1}v} \\ -\frac{B^{-1}v}{a-v'B^{-1}v} & B^{-1} + \frac{-B^{-1}vv'B^{-1}}{a-v'B^{-1}v} \end{bmatrix}$$

Notice that matrix B is the same for all regression, the identical terms for each regression are just a and v, which are very easy to calculate. So theoretically, we can reduce the cost of computation significantly.

5 Test

Now test two simulation examples by using the functions in this package.

```
> ## use fRLR package
> library(fRLR)

[1] "fRLR"      "stats"     "graphics"  "grDevices" "utils"     "datasets"
[7] "methods"   "base"

> set.seed(123)

NULL

> X = matrix(rnorm(50), 10, 5)

             [,1]       [,2]       [,3]        [,4]        [,5]
 [1,] -0.56047565  1.2240818 -1.0678237  0.42646422 -0.69470698
 [2,] -0.23017749  0.3598138 -0.2179749 -0.29507148 -0.20791728
 [3,]  1.55870831  0.4007715 -1.0260044  0.89512566 -1.26539635
 [4,]  0.07050839  0.1106827 -0.7288912  0.87813349  2.16895597
 [5,]  0.12928774 -0.5558411 -0.6250393  0.82158108  1.20796200
 [6,]  1.71506499  1.7869131 -1.6866933  0.68864025 -1.12310858
 [7,]  0.46091621  0.4978505  0.8377870  0.55391765 -0.40288484
 [8,] -1.26506123 -1.9666172  0.1533731 -0.06191171 -0.46665535
 [9,] -0.68685285  0.7013559 -1.1381369 -0.30596266  0.77996512
[10,] -0.44566197 -0.4727914  1.2538149 -0.38047100 -0.08336907

> Y = rnorm(10)

 [1]  0.25331851 -0.02854676 -0.04287046  1.36860228 -0.22577099  1.51647060
 [7] -1.54875280  0.58461375  0.12385424  0.21594157

> COV = matrix(rnorm(40), 10, 4)

             [,1]       [,2]         [,3]       [,4]
 [1,]  0.37963948 -0.4910312  0.005764186  0.9935039
 [2,] -0.50232345 -2.3091689  0.385280401  0.5483970
 [3,] -0.33320738  1.0057385 -0.370660032  0.2387317
 [4,] -1.01857538 -0.7092008  0.644376549 -0.6279061
```

```
 [5,] -1.07179123 -0.6880086 -0.220486562  1.3606524
 [6,]  0.30352864  1.0255714  0.331781964 -0.6002596
 [7,]  0.44820978 -0.2847730  1.096839013  2.1873330
 [8,]  0.05300423 -1.2207177  0.435181491  1.5326106
 [9,]  0.92226747  0.1813035 -0.325931586 -0.2357004
[10,]  2.05008469 -0.1388914  1.148807618 -1.0264209
```

```
> frlr1(X, Y, COV)
```

```
  r r.p.value
1 0 0.4380128
2 1 0.7791076
3 3 0.9495018
4 4 0.6729983
5 2 0.2212869
```

```
> ## use simple loop
> res = matrix(nrow = 0, ncol = 2)
```

```
     [,1] [,2]
```

```
> for (i in 1:ncol(X))
+ {
+   mat = cbind(X[,i], COV)
+   df = as.data.frame(mat)
+   model = lm(Y~., data = df)
+   tmp = c(i, summary(model)$coefficients[2, 4])
+   res = rbind(res, tmp)
+ }
```

```
NULL
```

```
> res
```

```
    [,1]      [,2]
tmp    1 0.4380128
tmp    2 0.7791076
tmp    3 0.2212869
tmp    4 0.9495018
tmp    5 0.6729983
```

As we can see in the above output, these p-values for the identical variable in each regression are equal betwwen two methods.

Similarly, we can test another example

```
> library(fRLR)
```

```
[1] "fRLR"      "stats"     "graphics"  "grDevices" "utils"     "datasets"
[7] "methods"   "base"
```

```
> set.seed(123)
```

```
NULL
```

```
> X = matrix(rnorm(50), 10, 5)

             [,1]       [,2]       [,3]        [,4]        [,5]
 [1,] -0.56047565  1.2240818 -1.0678237  0.42646422 -0.69470698
 [2,] -0.23017749  0.3598138 -0.2179749 -0.29507148 -0.20791728
 [3,]  1.55870831  0.4007715 -1.0260044  0.89512566 -1.26539635
 [4,]  0.07050839  0.1106827 -0.7288912  0.87813349  2.16895597
 [5,]  0.12928774 -0.5558411 -0.6250393  0.82158108  1.20796200
 [6,]  1.71506499  1.7869131 -1.6866933  0.68864025 -1.12310858
 [7,]  0.46091621  0.4978505  0.8377870  0.55391765 -0.40288484
 [8,] -1.26506123 -1.9666172  0.1533731 -0.06191171 -0.46665535
 [9,] -0.68685285  0.7013559 -1.1381369 -0.30596266  0.77996512
[10,] -0.44566197 -0.4727914  1.2538149 -0.38047100 -0.08336907

> Y = rnorm(10)

 [1]  0.25331851 -0.02854676 -0.04287046  1.36860228 -0.22577099  1.51647060
 [7] -1.54875280  0.58461375  0.12385424  0.21594157

> COV = matrix(rnorm(40), 10, 4)

             [,1]       [,2]         [,3]       [,4]
 [1,]  0.37963948 -0.4910312  0.005764186  0.9935039
 [2,] -0.50232345 -2.3091689  0.385280401  0.5483970
 [3,] -0.33320738  1.0057385 -0.370660032  0.2387317
 [4,] -1.01857538 -0.7092008  0.644376549 -0.6279061
 [5,] -1.07179123 -0.6880086 -0.220486562  1.3606524
 [6,]  0.30352864  1.0255714  0.331781964 -0.6002596
 [7,]  0.44820978 -0.2847730  1.096839013  2.1873330
 [8,]  0.05300423 -1.2207177  0.435181491  1.5326106
 [9,]  0.92226747  0.1813035 -0.325931586 -0.2357004
[10,]  2.05008469 -0.1388914  1.148807618 -1.0264209

> idx1 = c(1, 2, 3, 4, 1, 1, 1, 2, 2, 3)

 [1] 1 2 3 4 1 1 1 2 2 3

> idx2 = c(2, 3, 4, 5, 3, 4, 5, 4, 5, 5)

 [1] 2 3 4 5 3 4 5 4 5 5

> frlr2(X, idx1, idx2, Y, COV)

   r1 r2 r1.p.value  r2.p.value
1   1  2 0.53021406 0.895719578
2   2  3 0.01812006 0.009833047
3   4  5 0.91749181 0.712075464
4   1  5 0.12479380 0.152802911
5   3  4 0.29895922 0.963995969
6   2  4 0.79302893 0.902402294
7   1  3 0.33761507 0.210331456
8   2  5 0.73153760 0.663392258
9   3  5 0.32367303 0.877154122
10  1  4 0.51074586 0.966484642
```

```
> res = matrix(nrow=0, ncol=4)

     [,1] [,2] [,3] [,4]

> for (i in 1:length(idx1))
+ {
+   mat = cbind(X[, idx1[i]], X[,idx2[i]], COV)
+   df = as.data.frame(mat)
+   model = lm(Y~., data = df)
+   tmp = c(idx1[i], idx2[i], summary(model)$coefficients[2,4], summary(model)$coefficients[3,4
+   res = rbind(res, tmp)
+ }

NULL
```

Again, we obtain the same results by different methods.

6 Computation Performance

The main aim of this new method is to reduce the computation cost. Now let's compare its speed with the simple-loop method.

We can obtain the following time cost for $99 \times 100/2 = 4950$ linear regressions.

```
> library(fRLR)
> set.seed(123)
> n = 100
> X = matrix(rnorm(10*n), 10, n)
> Y = rnorm(10)
> COV = matrix(rnorm(40), 10, 4)
> #idx1 = c(1, 2, 3, 4, 1, 1, 1, 2, 2, 3)
> #idx2 = c(2, 3, 4, 5, 3, 4, 5, 4, 5, 5)
> id = combn(n, 2)
> idx1 = id[1, ]
> idx2 = id[2, ]
> system.time(frlr2(X, idx1, idx2, Y, COV))

   user  system elapsed
  0.052   0.000   0.014

> simpleLoop <- function()
+ {
+   res = matrix(nrow=0, ncol=4)
+   for (i in 1:length(idx1))
+   {
+     mat = cbind(X[, idx1[i]], X[,idx2[i]], COV)
+     df = as.data.frame(mat)
+     model = lm(Y~., data = df)
+     tmp = c(idx1[i], idx2[i], summary(model)$coefficients[2,4], summary(model)$coefficients[3
+     res = rbind(res, tmp)
+   }
+ }
> system.time(simpleLoop())
```

```
  user  system elapsed 
 5.976   0.008   5.986 
```