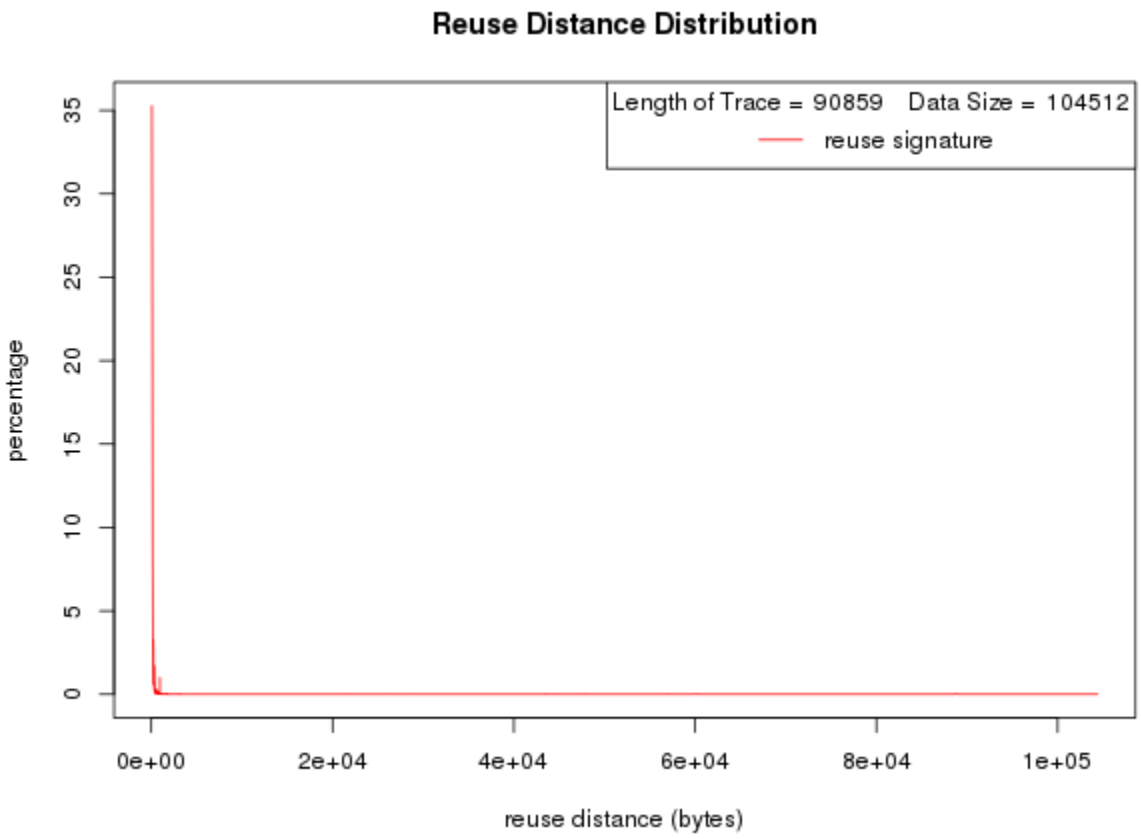
Dan Scarafoni and Praneet Tata (dscarafo and mtata)
Locality lab
csc252

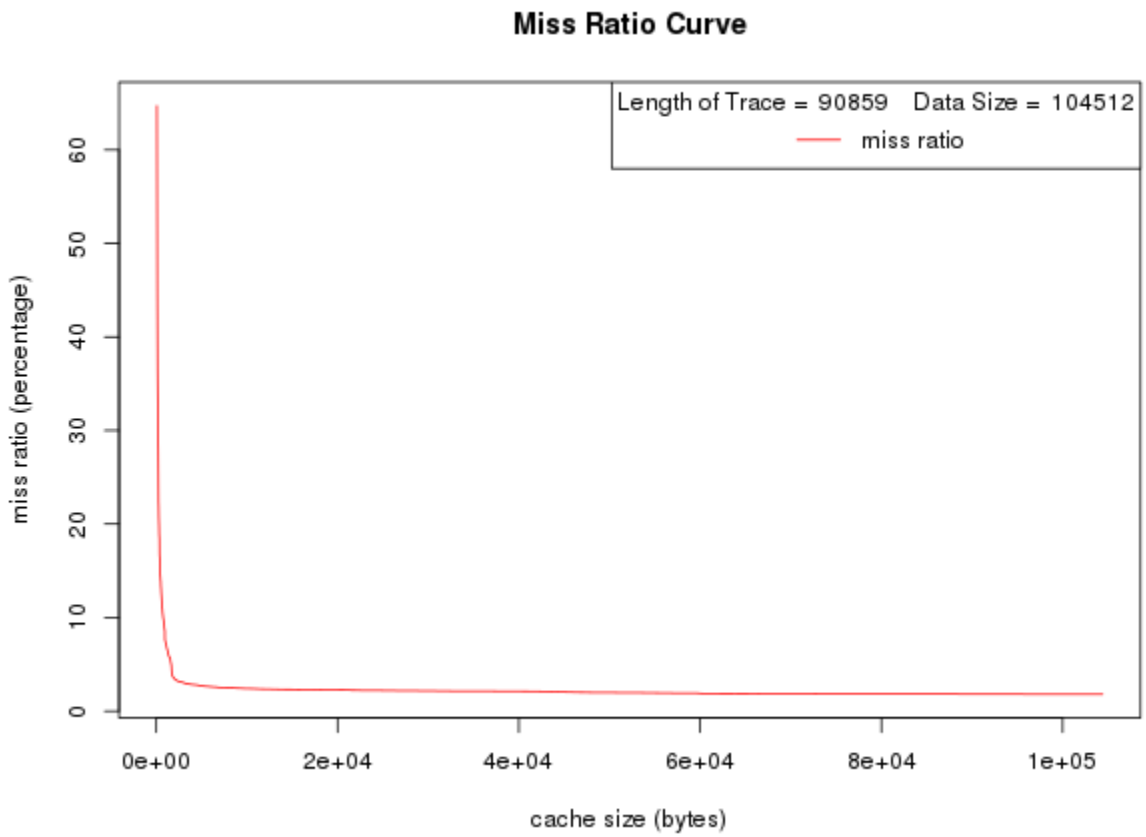
Our lab sought to understand the effects of compiler optimization and tiling on the run time and locality of functions. For our experiments, the program used was a simple matrix multiplication algorithm written in c. This program allowed the size of the multiplied matrices to be decided at run time. A separate copy of the function used for tiling functions could allowed tiling of various sizes of the matrices (the exact amount was determined by a command line argument).

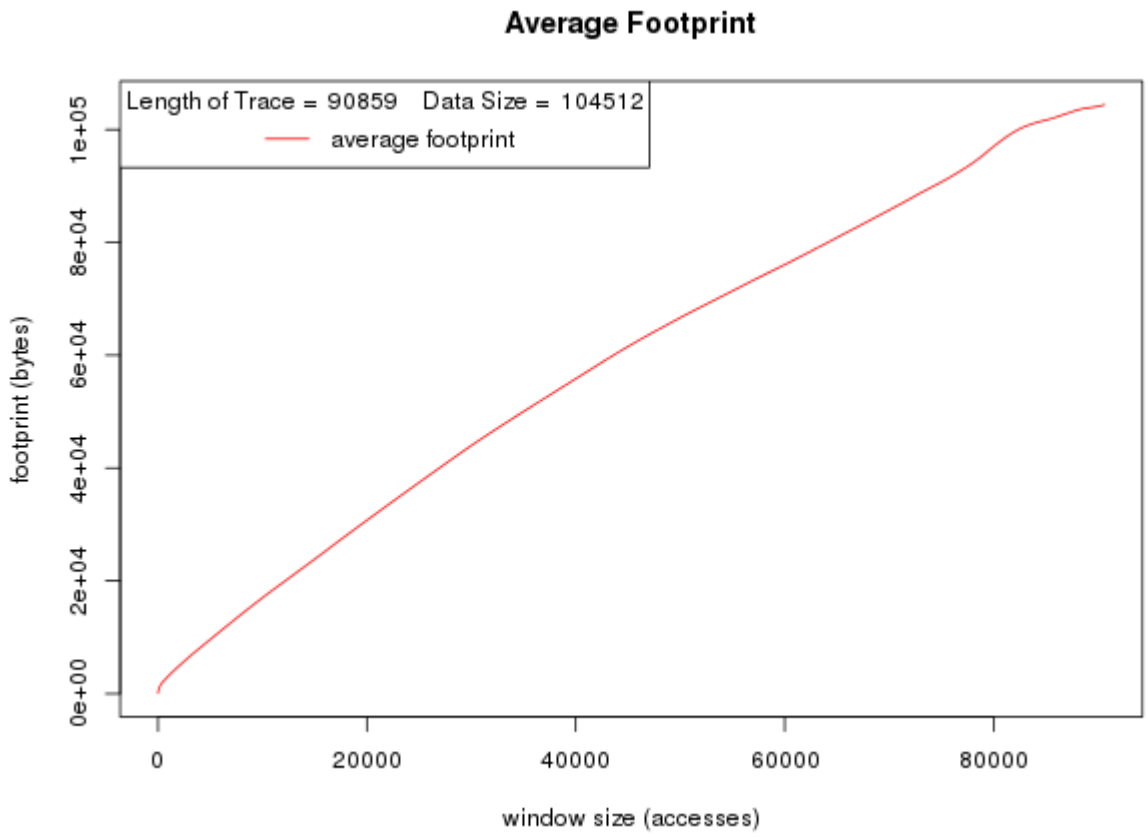
The run time of the program is defined as the time needed to execute the entire program, in milliseconds, from start to finish. The definition of locality (as is relevant for this experiment) is slightly more complicated. The reuse distance of a memory access is the amount of unique data items used in between it and the previous access to the data. Similarly, the reuse time is the amount of total data used in between access of the same piece of information. The average footprint is a combination between the two, and is the amount of data accessed over a particular period of time (time window). The miss ratio is a measure of the ratio of attempts at retrieving information not present in the cache to those present within. More formally, it is fraction of reuse distances greater than the cache size.

For this experiment, three major measures of locality: reuse distance, miss ratio, and average footprint. The program Loca was used to gather locality data from. Further note that all experiments were run on a machine with a fully associative cache with a least-recently-used replacement policy.

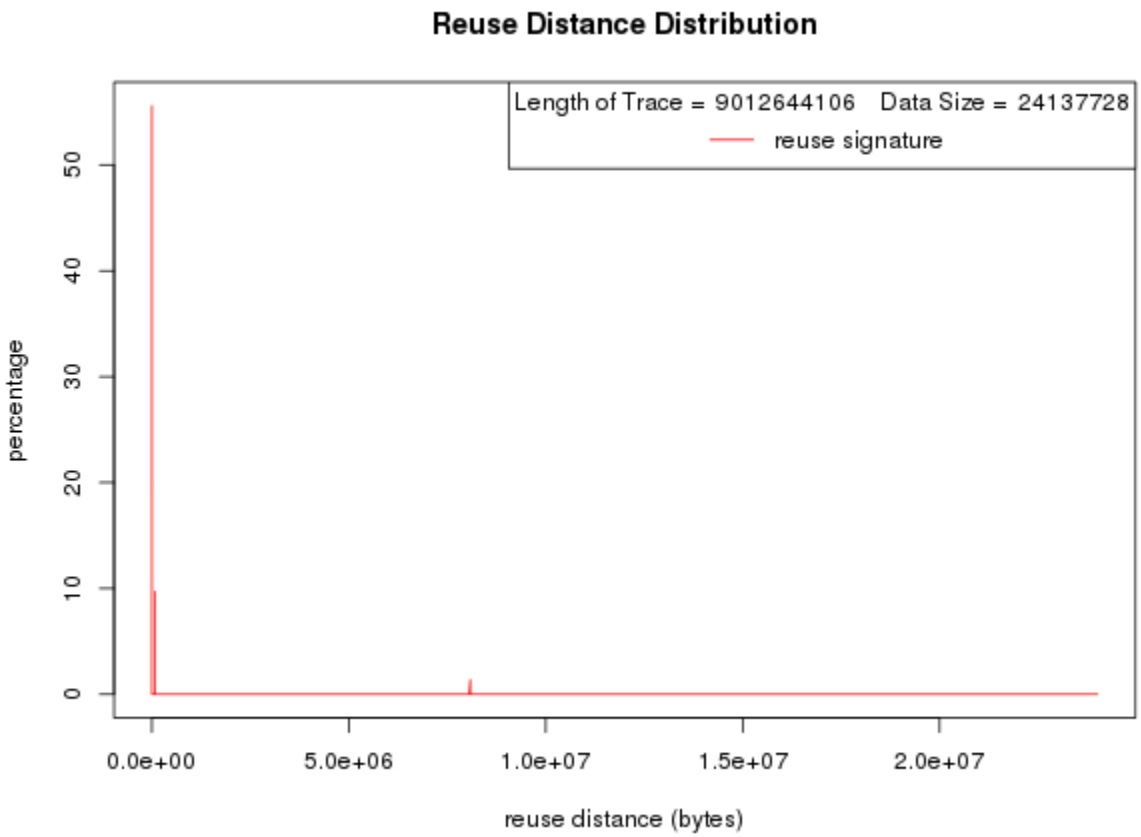
The first experiment noted the effect of compiler optimization on locality and run time, the following graphs demonstrate the relevant findings for various magnitudes of compiler optimization (created with the gcc flag -Ox, where x is an integer between 0 and 3). These optimization setting were in turn experimented with with matrices of sizes 10x10 and 1000x1000.

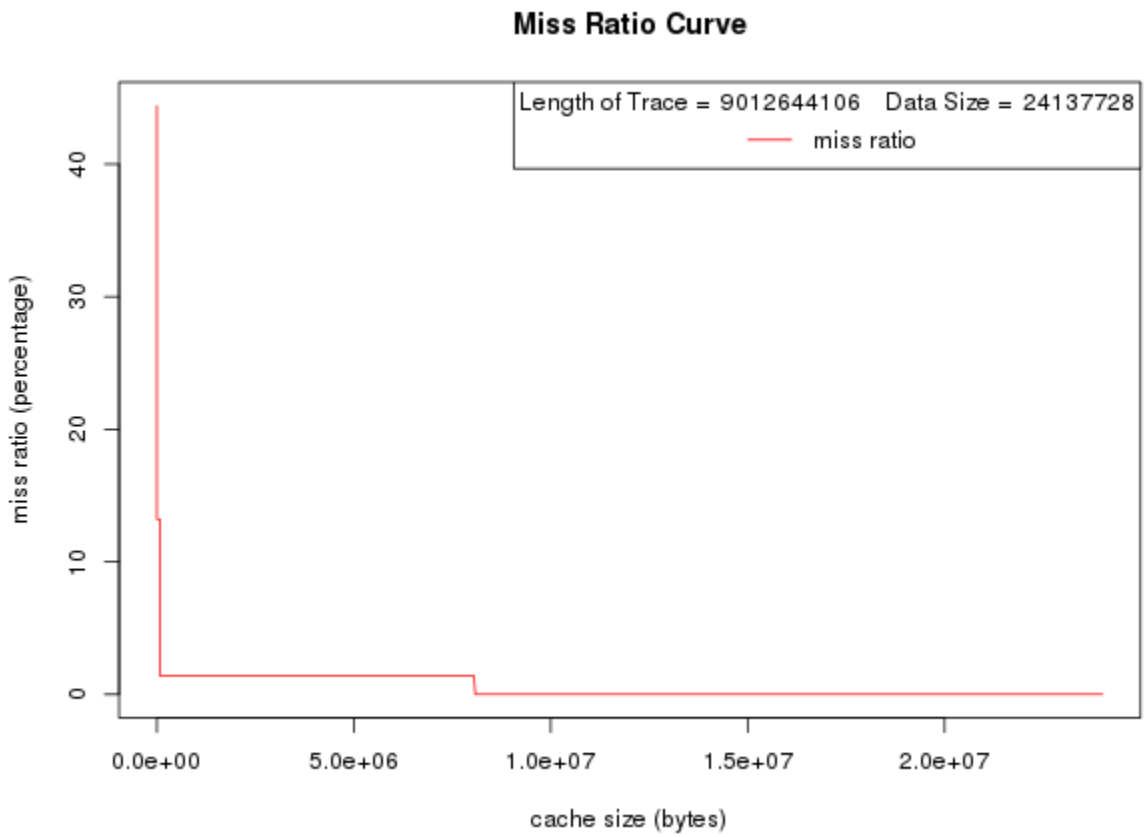

Reuse Distance Distribution
Length of Trace = 90859 Data Size = 104512
reuse signature
percentage
0
5
10
15
20
25
30
35
0e+00
2e+04
4e+04
6e+04
8e+04
1e+05
reuse distance (bytes)

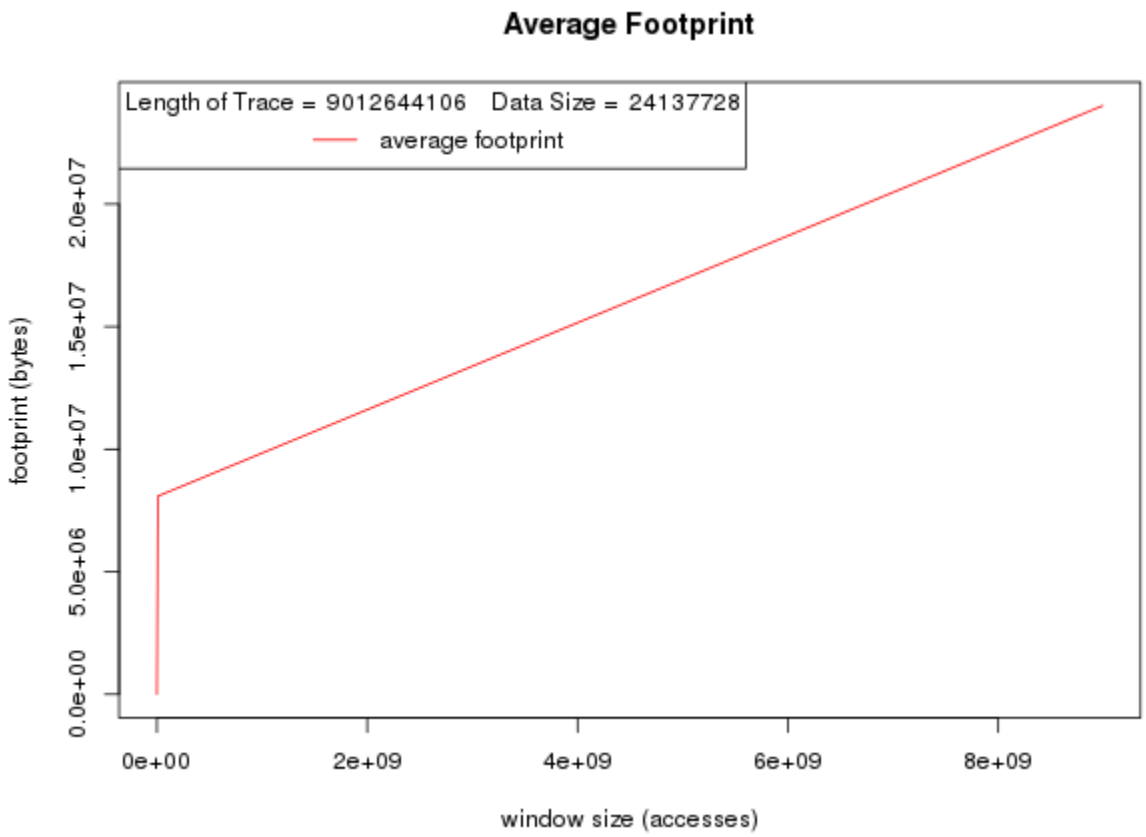

Miss Ratio Curve
Length of Trace = 90859 Data Size = 104512
miss ratio
miss ratio (percentage)
0
10
20
30
40
50
60
0e+00
2e+04
4e+04
6e+04
8e+04
1e+05
cache size (bytes)



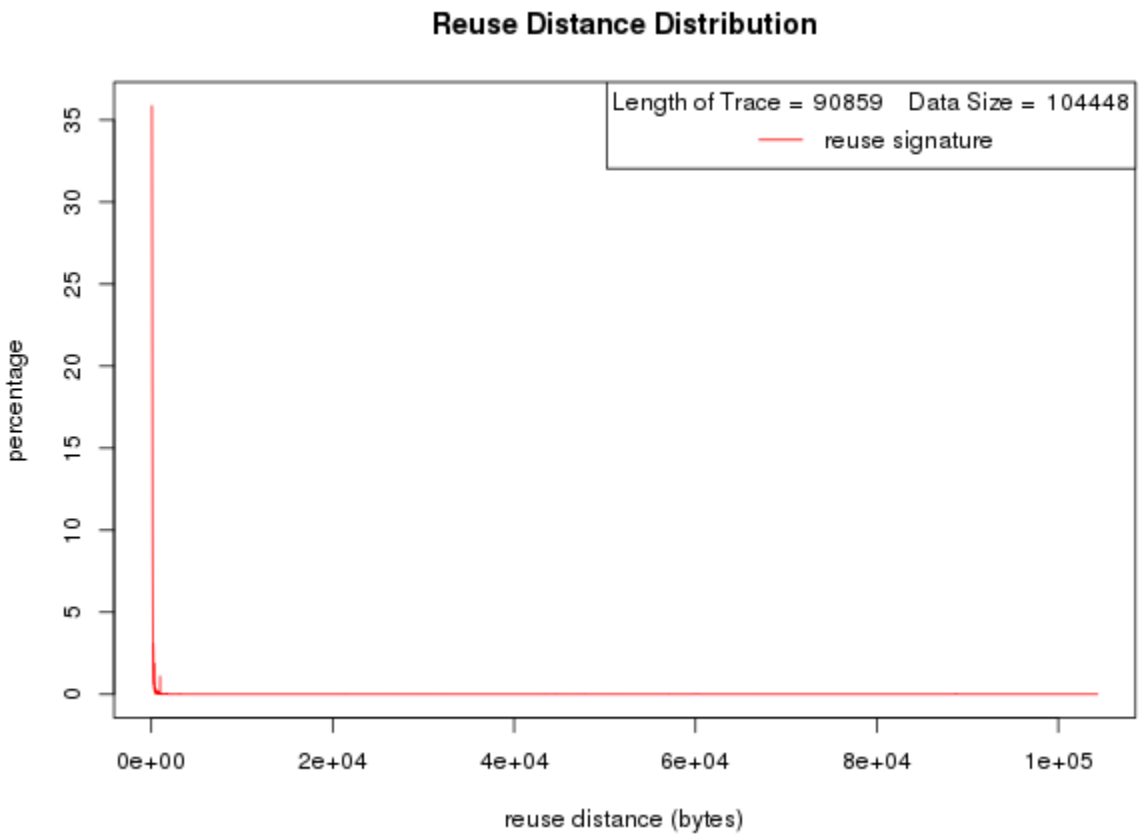
Opt0 size 10
time- 2.8

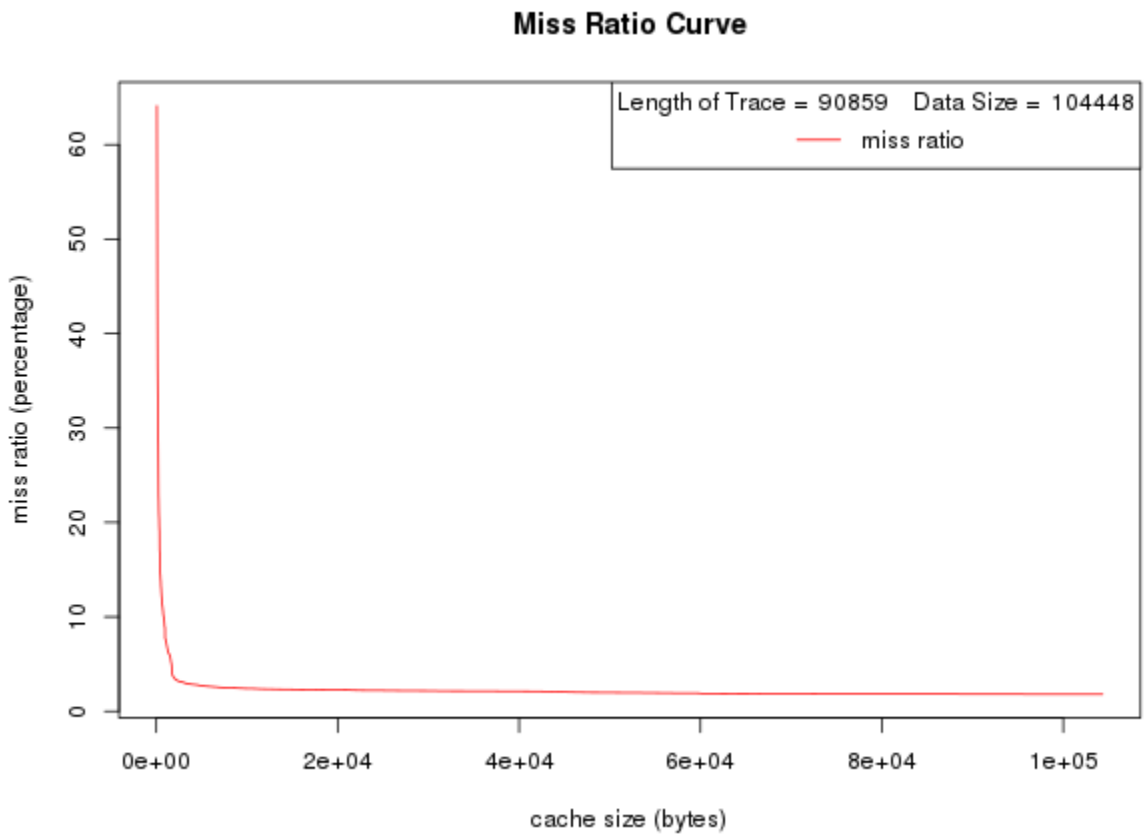

Reuse Distance Distribution
Length of Trace = 9012644106 Data Size = 24137728
reuse signature
percentage
0
10
20
30
40
50
0.0e+00
5.0e+06
1.0e+07
1.5e+07
2.0e+07
reuse distance (bytes)

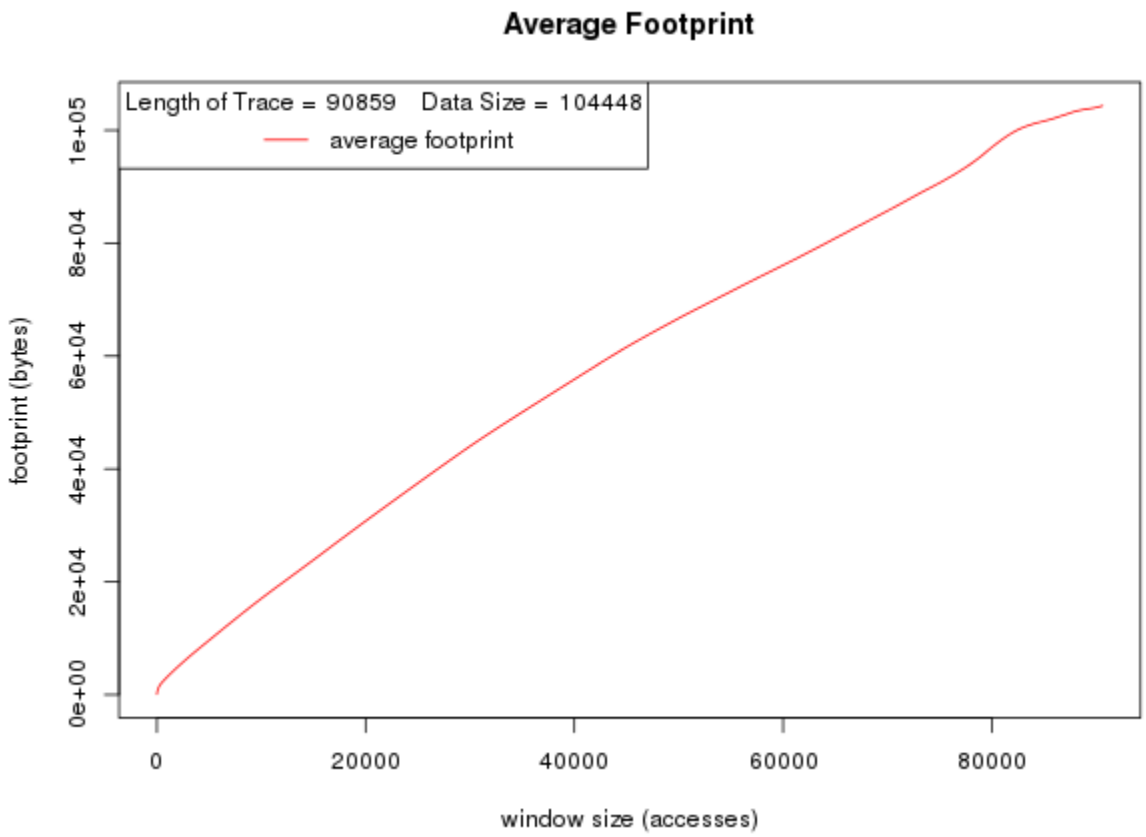

Miss Ratio Curve
Length of Trace = 9012644106 Data Size = 24137728
miss ratio
miss ratio (percentage)
0
10
20
30
40
0.0e+00
5.0e+06
1.0e+07
1.5e+07
2.0e+07
cache size (bytes)



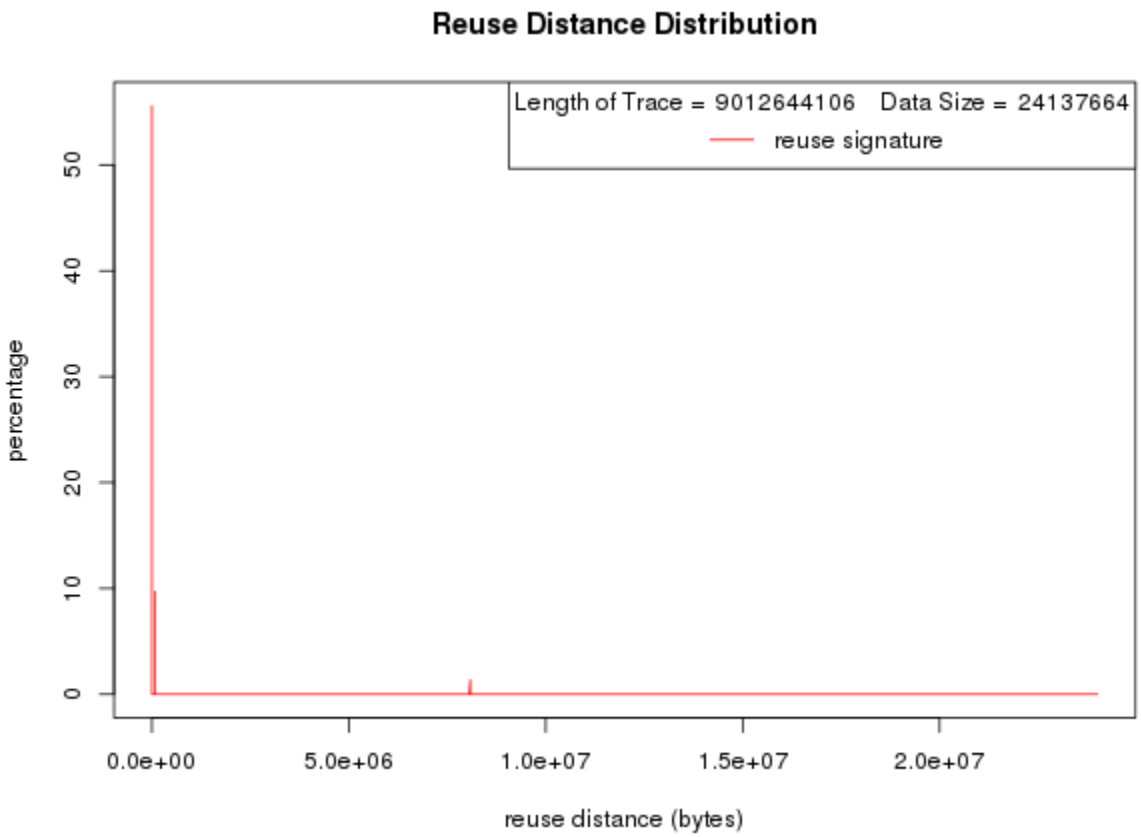
opt0 size 1000
time- 722.65

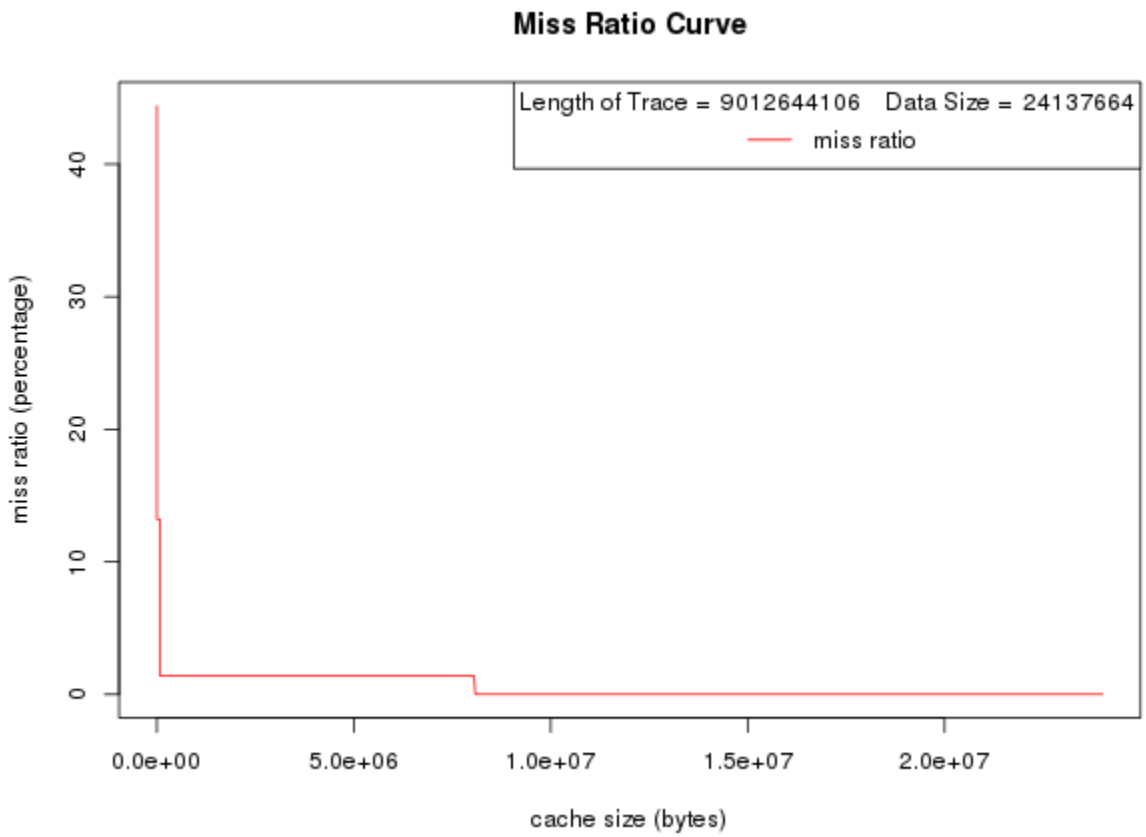

Reuse Distance Distribution
Length of Trace = 90859 Data Size = 104448
reuse signature
percentage
0
5
10
15
20
25
30
35
0e+00
2e+04
4e+04
6e+04
8e+04
1e+05
reuse distance (bytes)

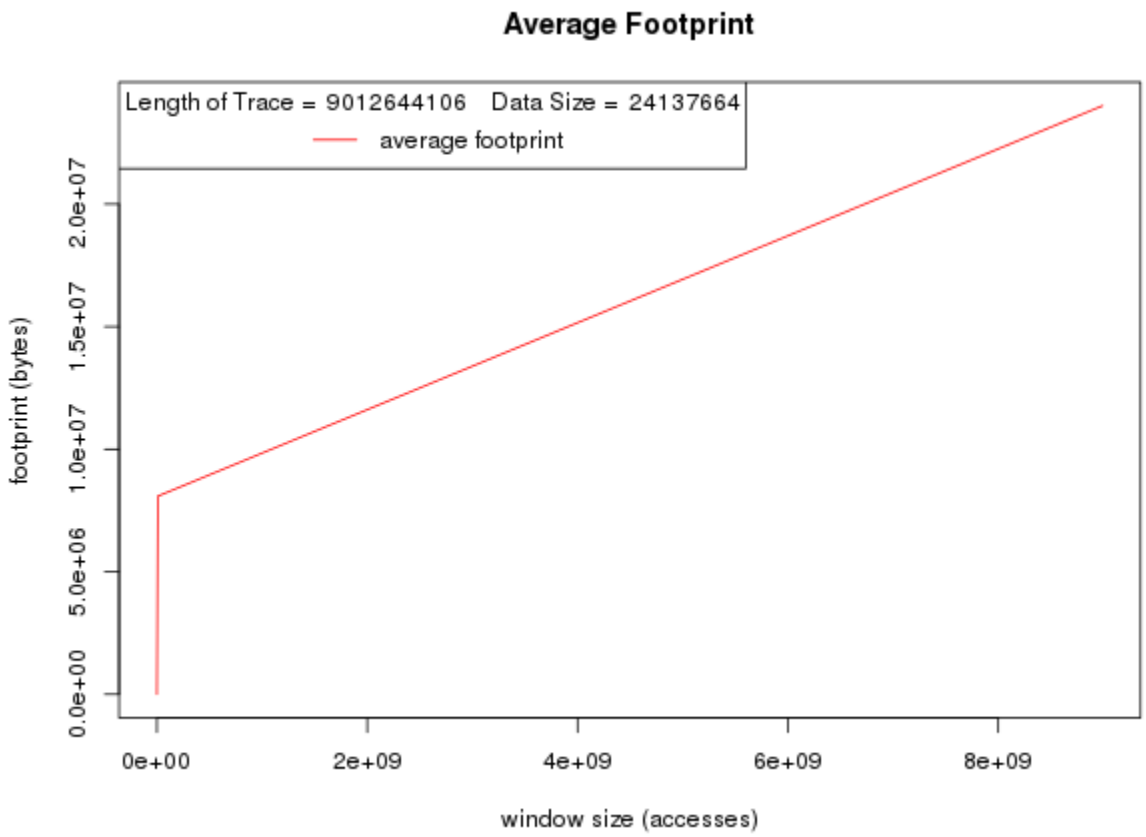

Miss Ratio Curve
Length of Trace = 90859 Data Size = 104448
miss ratio
miss ratio (percentage)
0
10
20
30
40
50
60
0e+00
2e+04
4e+04
6e+04
8e+04
1e+05
cache size (bytes)



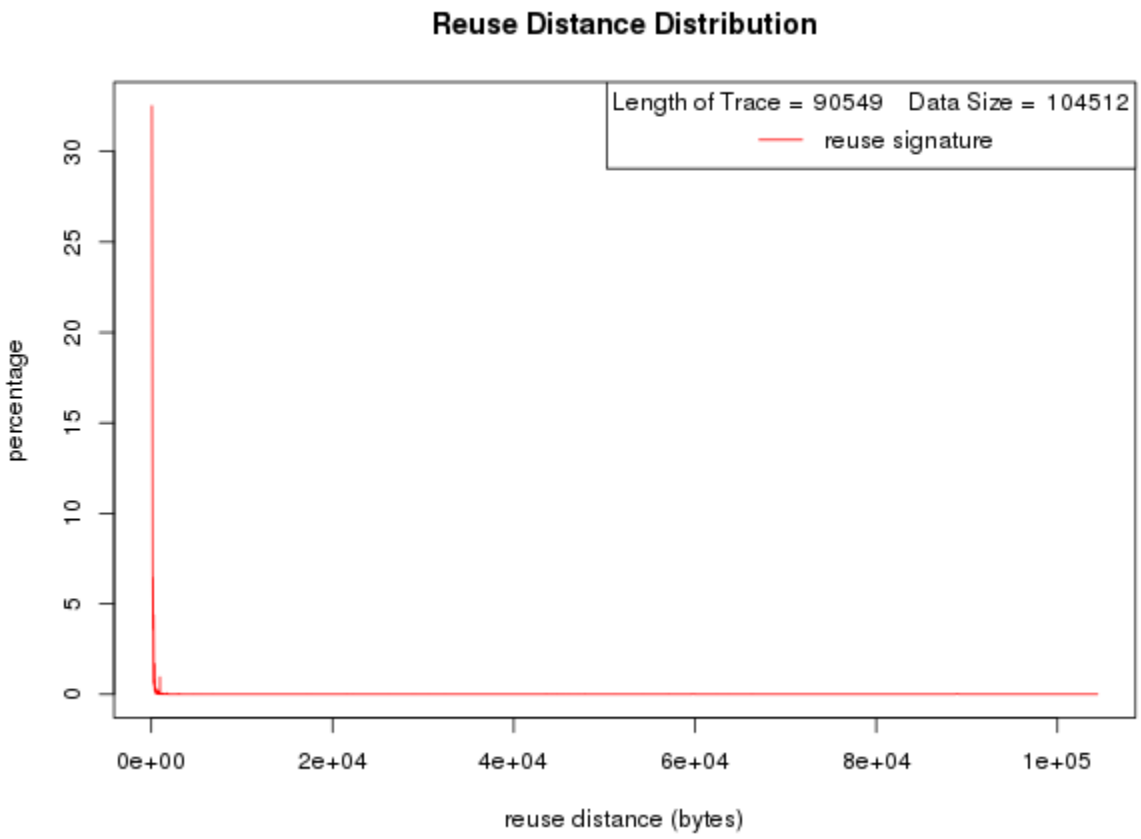
opt1 size 10
time- 2.92

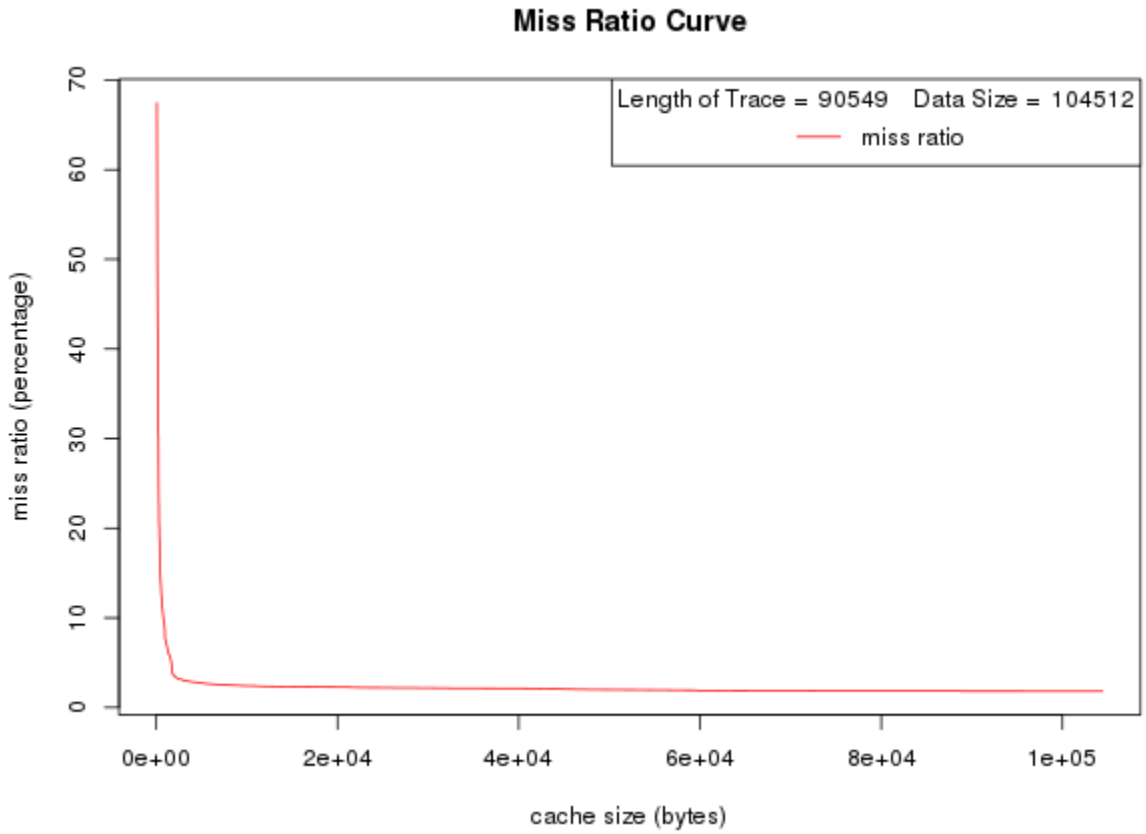

Reuse Distance Distribution
Length of Trace = 9012644106 Data Size = 24137664
reuse signature
percentage
0
10
20
30
40
50
0.0e+00
5.0e+06
1.0e+07
1.5e+07
2.0e+07
reuse distance (bytes)

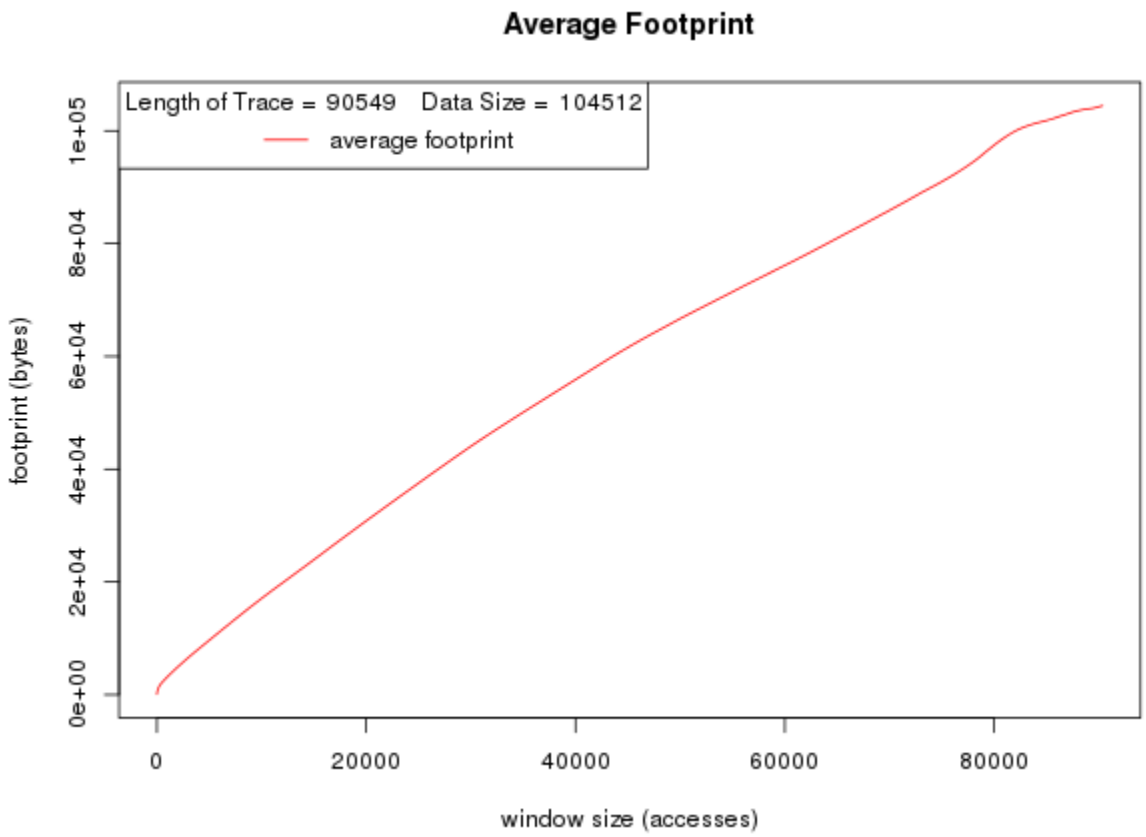

Miss Ratio Curve
Length of Trace = 9012644106 Data Size = 24137664
miss ratio
miss ratio (percentage)
0
10
20
30
40
0.0e+00
5.0e+06
1.0e+07
1.5e+07
2.0e+07
cache size (bytes)



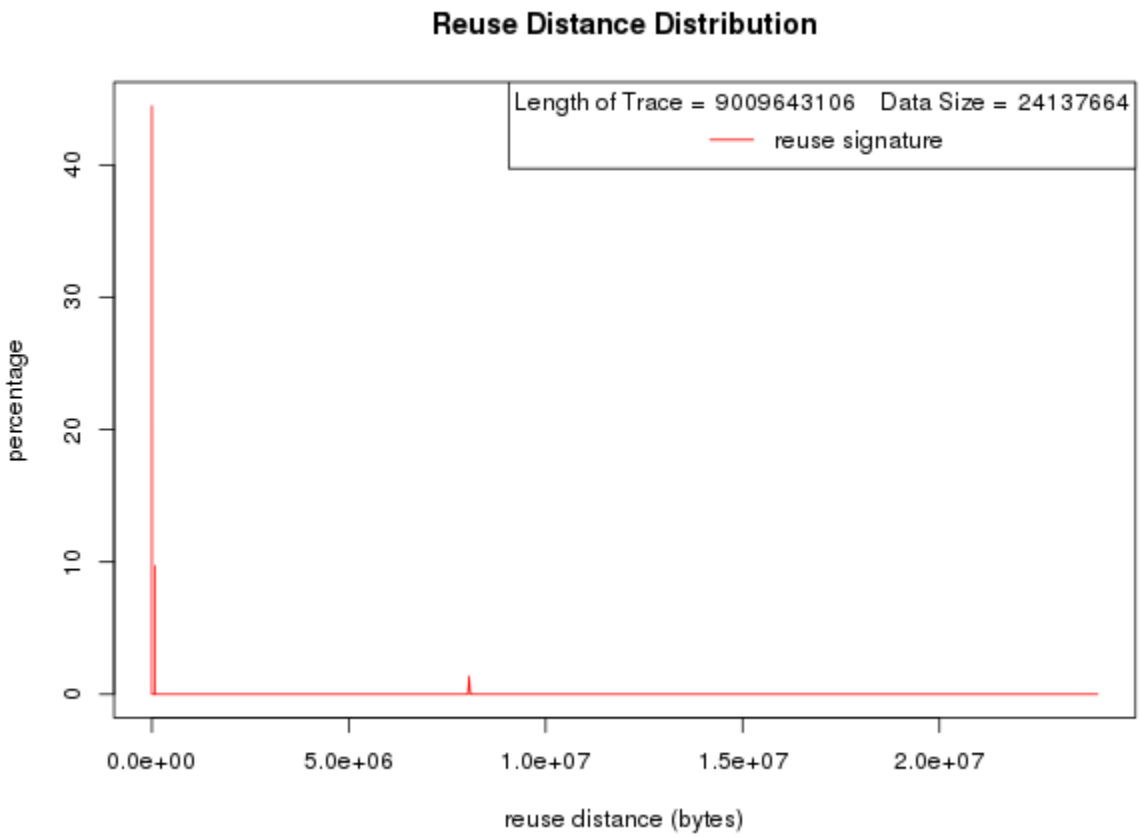
opt1 size 1000
time- 728.2

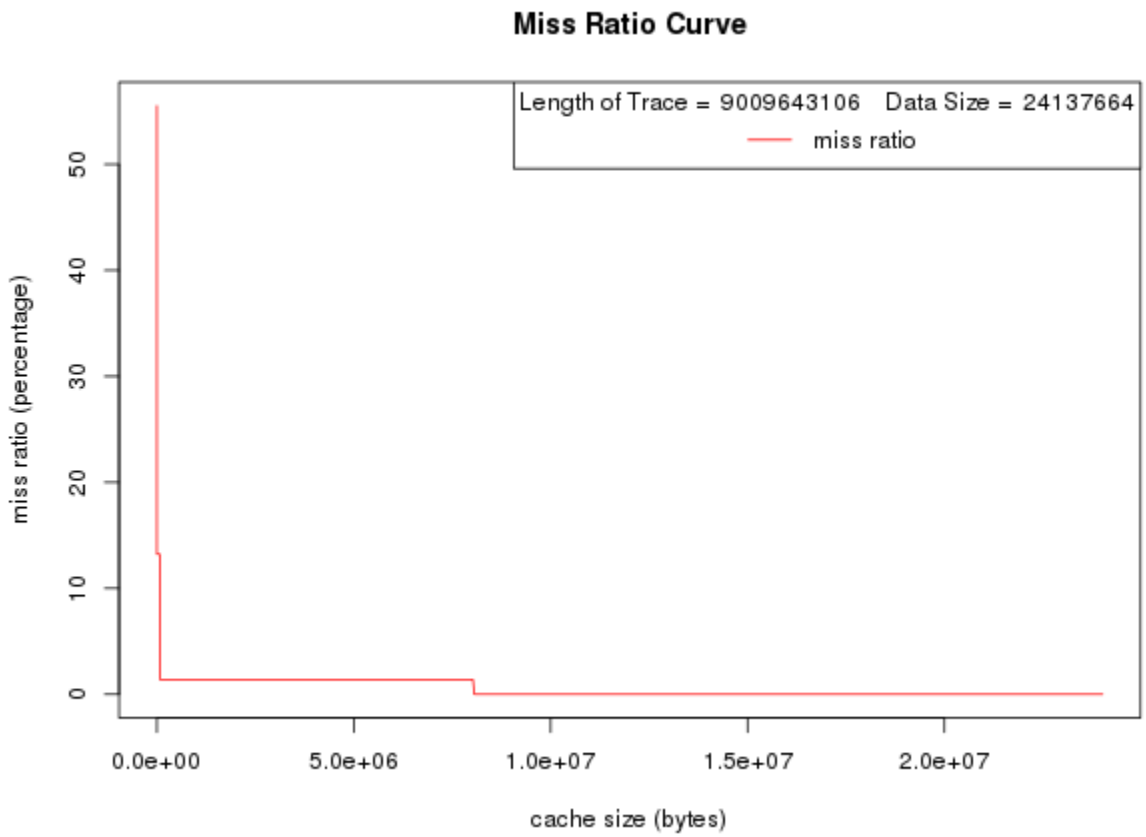

Reuse Distance Distribution
Length of Trace = 90549 Data Size = 104512
reuse signature
percentage
0
5
10
15
20
25
30
0e+00
2e+04
4e+04
6e+04
8e+04
1e+05
reuse distance (bytes)

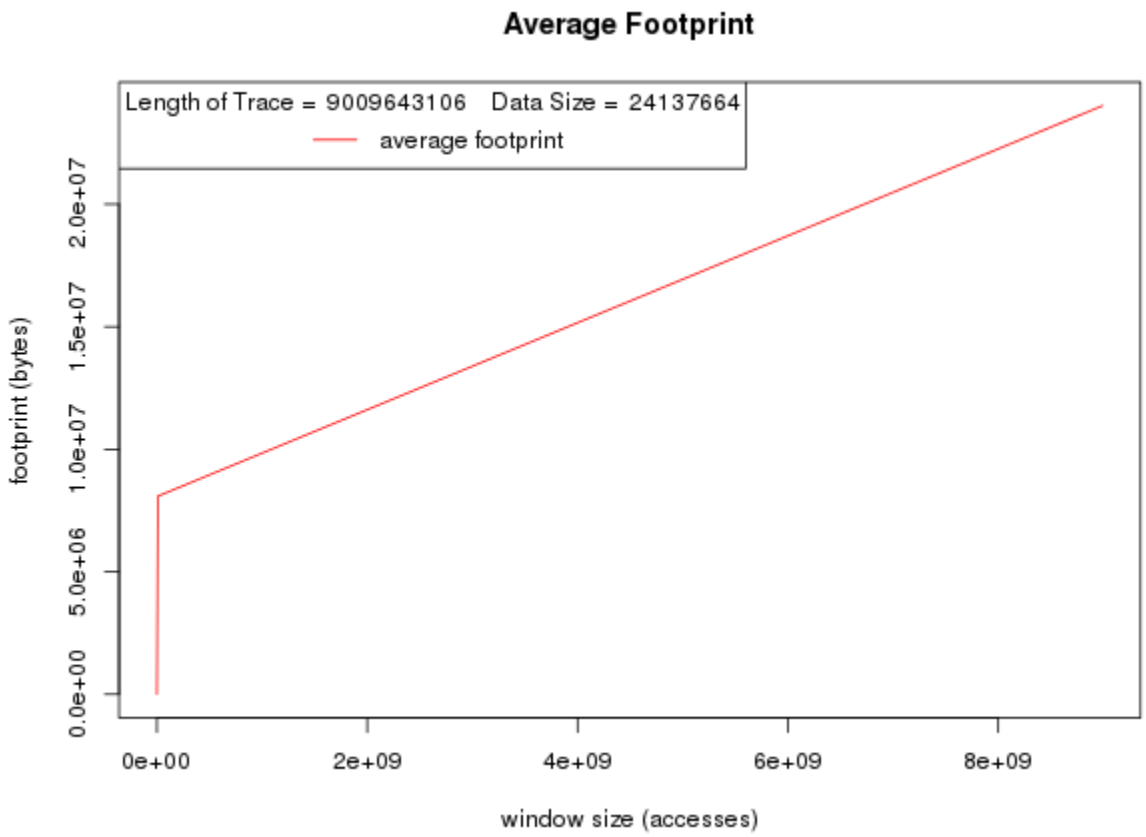

Miss Ratio Curve
Length of Trace = 90549 Data Size = 104512
miss ratio
miss ratio (percentage)
0
10
20
30
40
50
60
70
0e+00
2e+04
4e+04
6e+04
8e+04
1e+05
cache size (bytes)



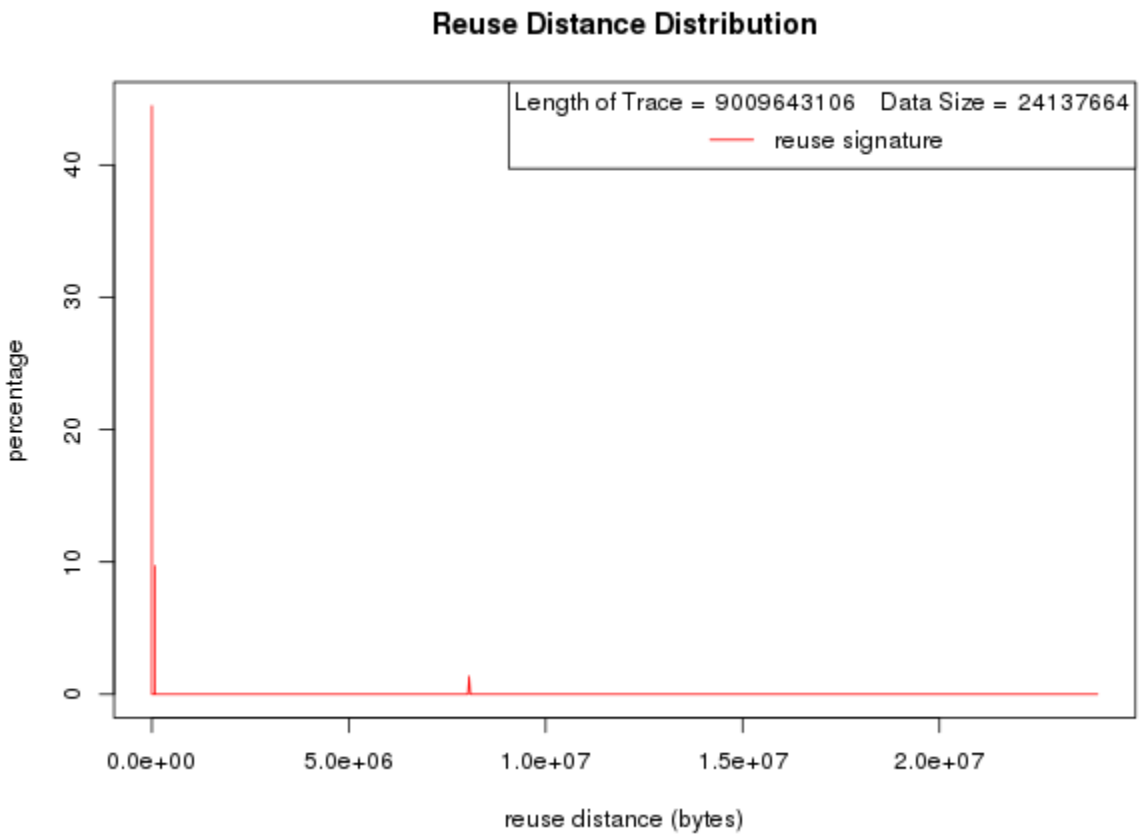
opt2 size 10
time- 5.11

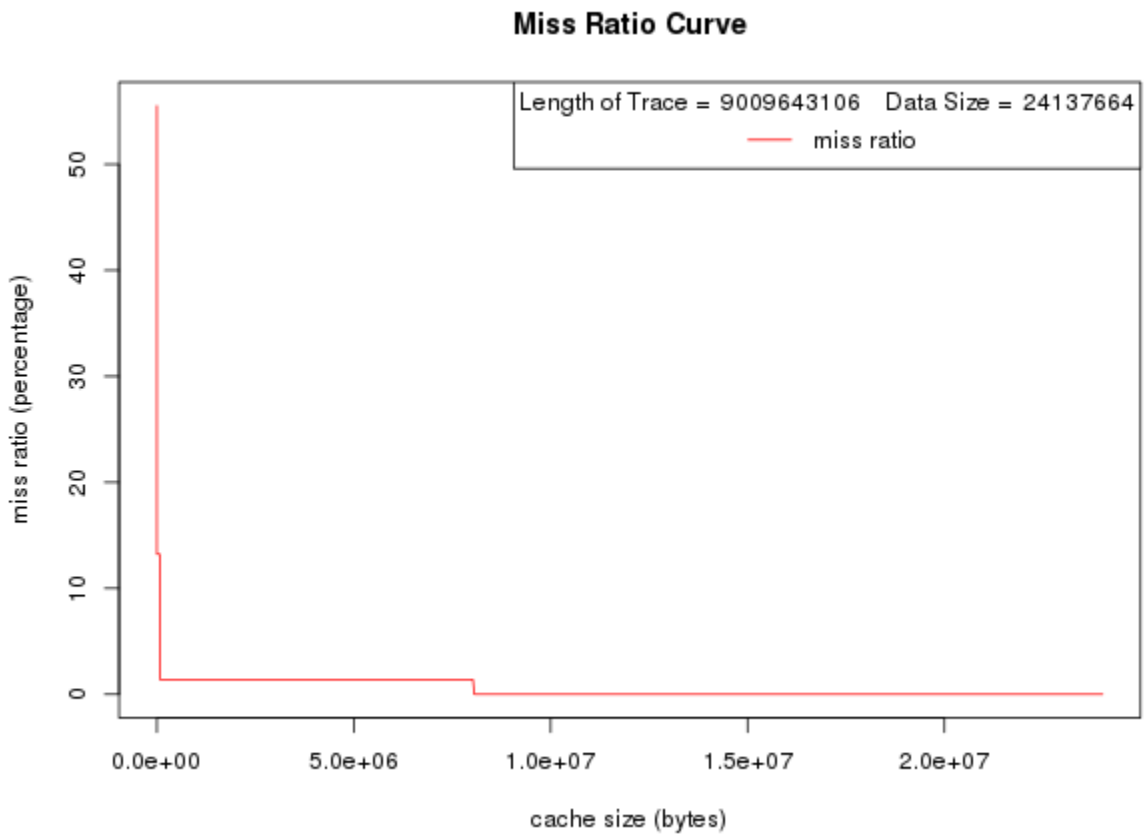

Reuse Distance Distribution
Length of Trace = 9009643106 Data Size = 24137664
reuse signature
percentage
0
10
20
30
40
0.0e+00
5.0e+06
1.0e+07
1.5e+07
2.0e+07
reuse distance (bytes)

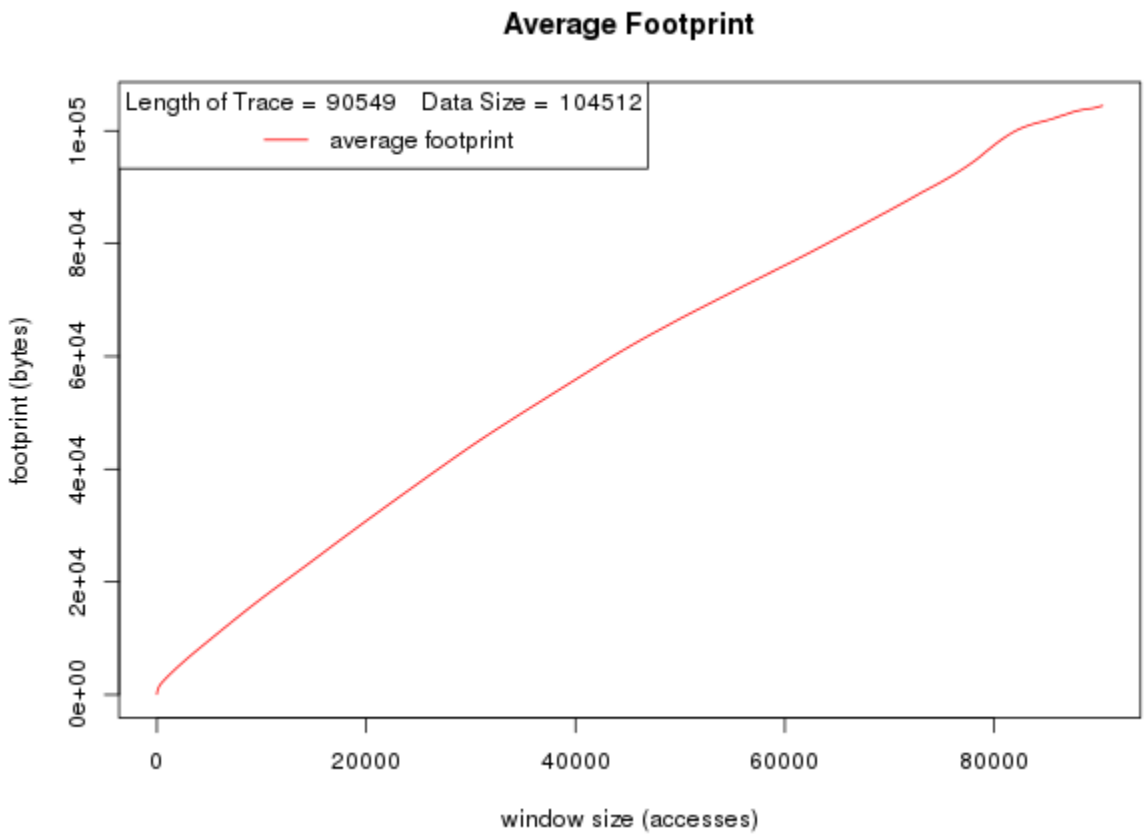

Miss Ratio Curve
Length of Trace = 9009643106 Data Size = 24137664
miss ratio
miss ratio (percentage)
0
10
20
30
40
50
0.0e+00
5.0e+06
1.0e+07
1.5e+07
2.0e+07
cache size (bytes)



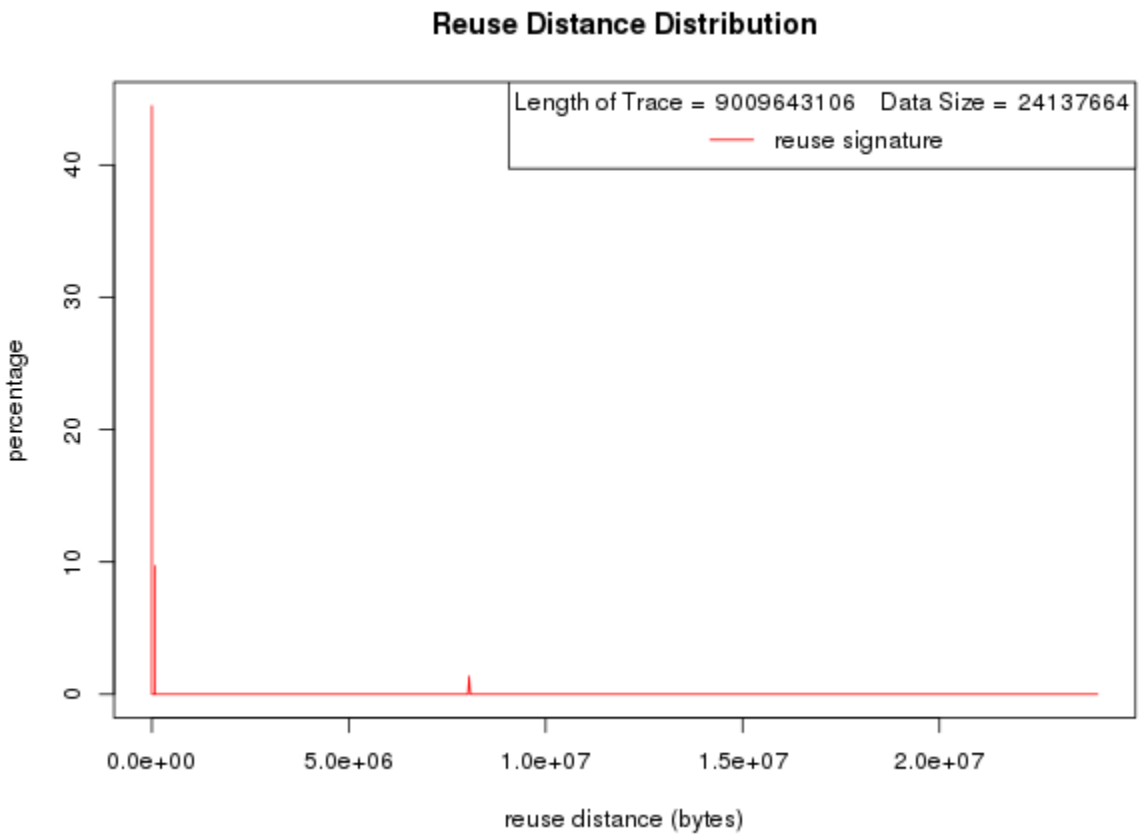
opt2 size 1000
time- 731.43

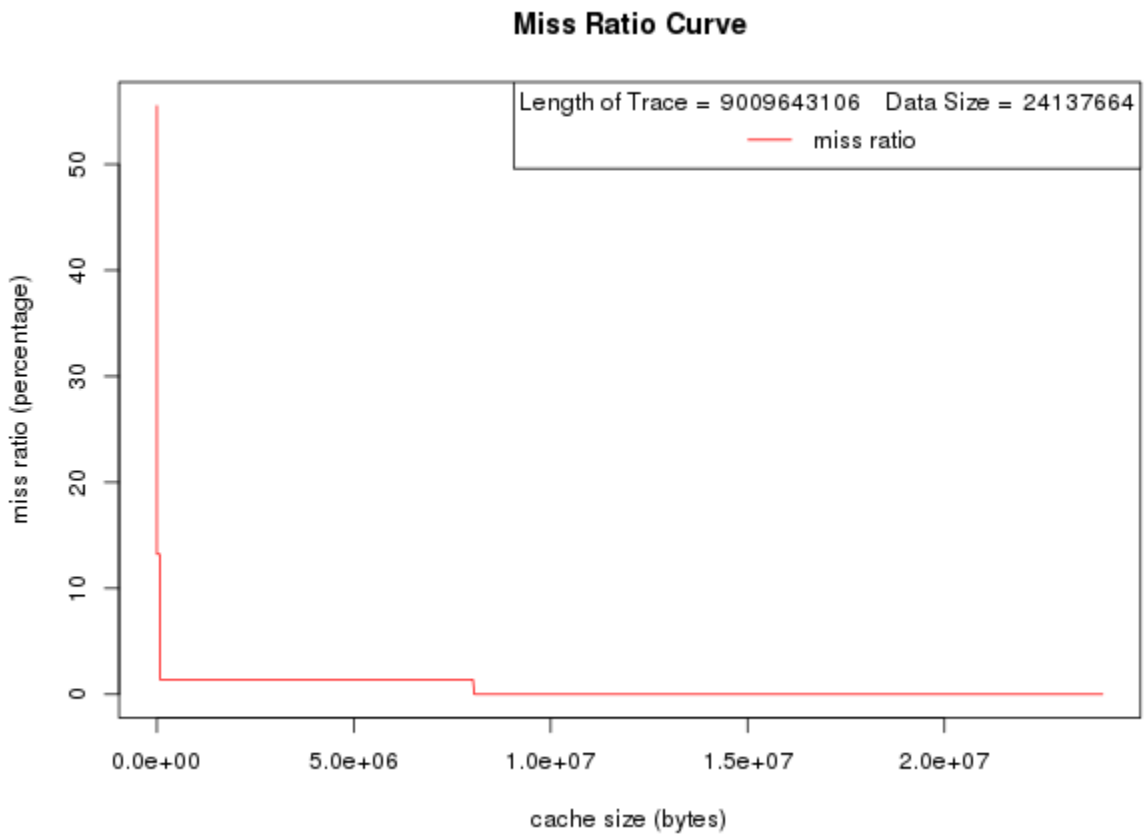

Reuse Distance Distribution
Length of Trace = 9009643106 Data Size = 24137664
reuse signature
percentage
0
10
20
30
40
0.0e+00
5.0e+06
1.0e+07
1.5e+07
2.0e+07
reuse distance (bytes)

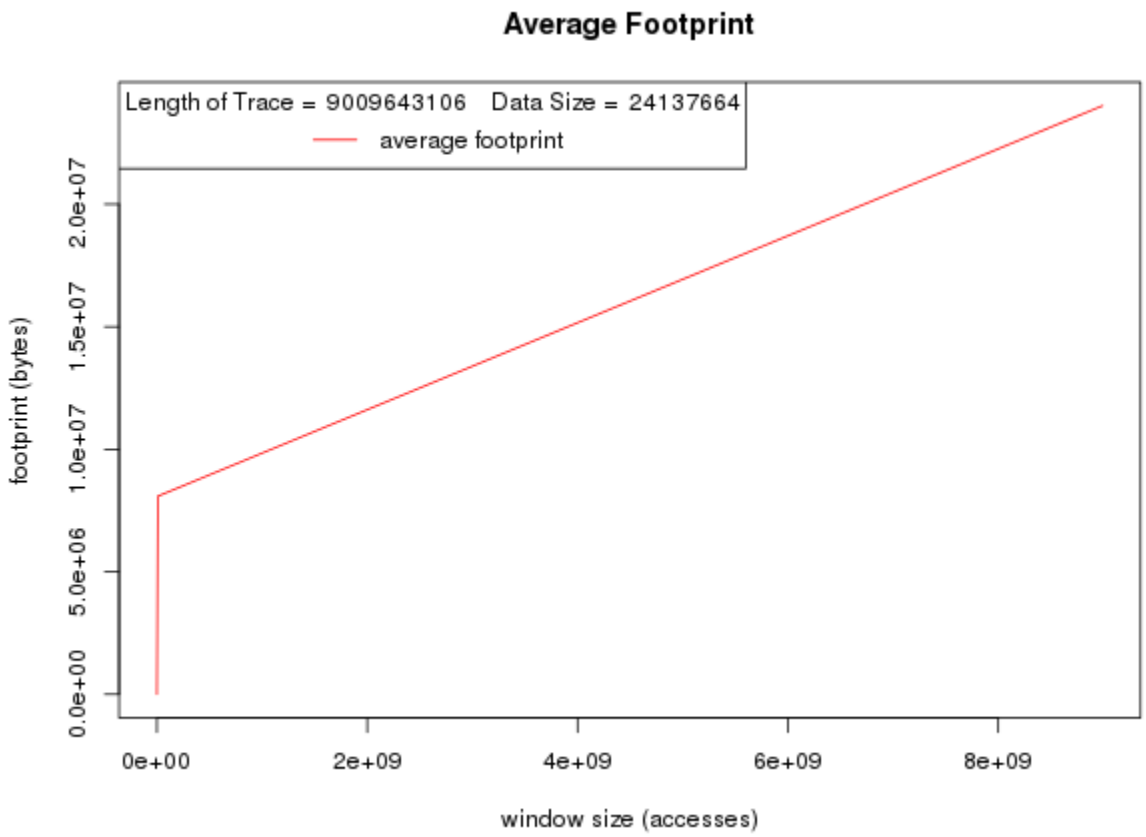

Miss Ratio Curve
Length of Trace = 9009643106 Data Size = 24137664
miss ratio
miss ratio (percentage)
0
10
20
30
40
50
0.0e+00
5.0e+06
1.0e+07
1.5e+07
2.0e+07
cache size (bytes)



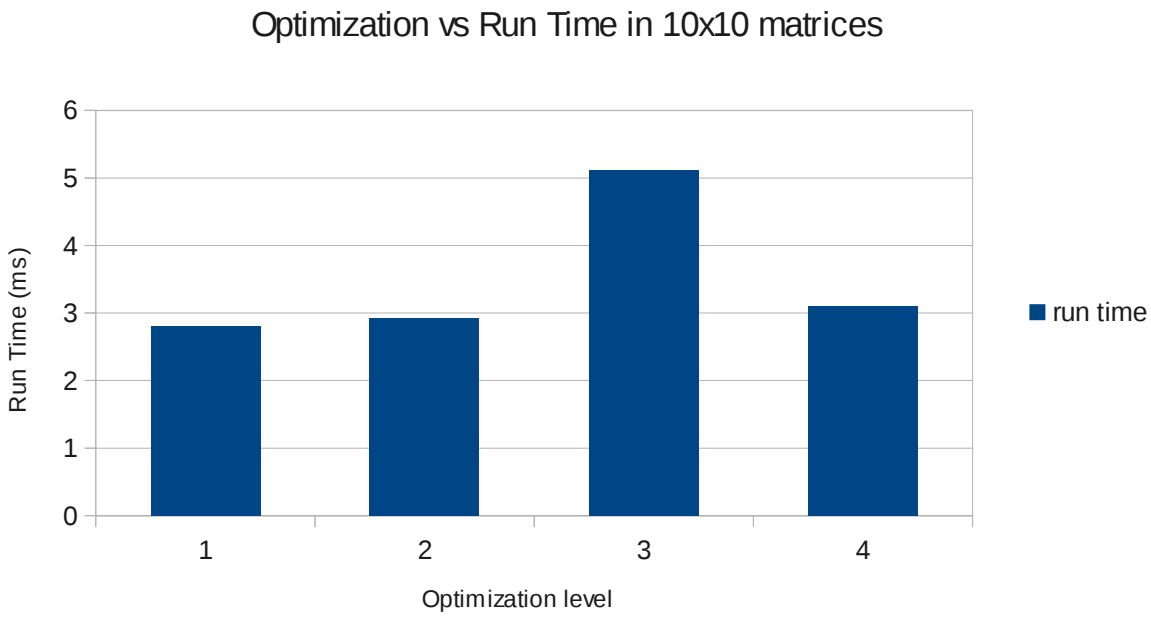
opt3 size 10
time- 3.10

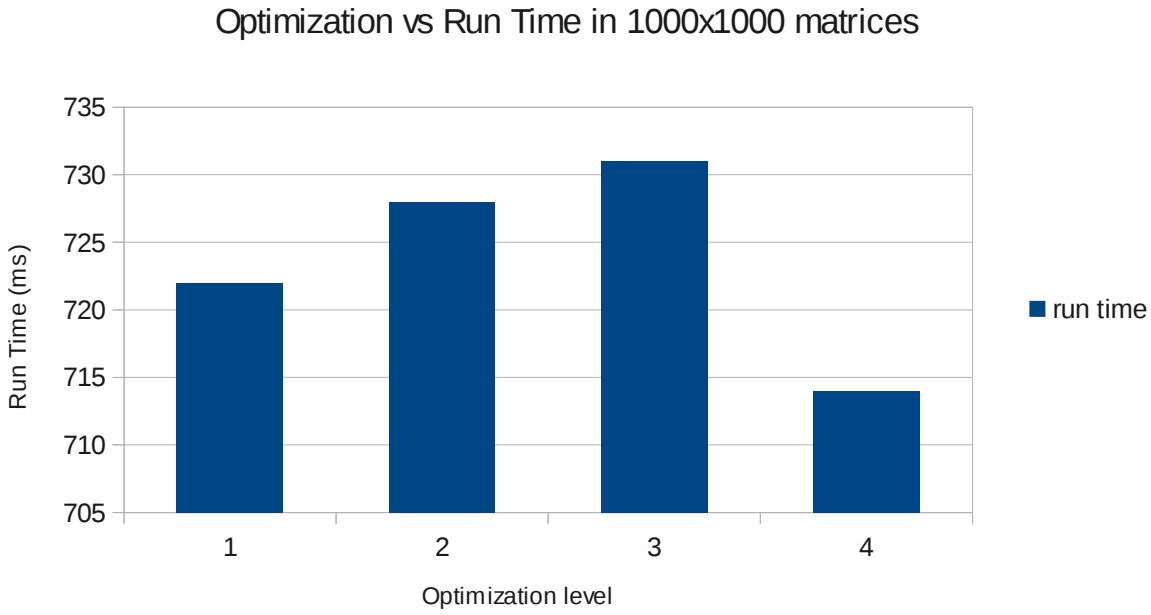

Reuse Distance Distribution
Length of Trace = 9009643106 Data Size = 24137664
reuse signature
percentage
0
10
20
30
40
0.0e+00
5.0e+06
1.0e+07
1.5e+07
2.0e+07
reuse distance (bytes)


Miss Ratio Curve
Length of Trace = 9009643106 Data Size = 24137664
miss ratio
miss ratio (percentage)
0
10
20
30
40
50
0.0e+00
5.0e+06
1.0e+07
1.5e+07
2.0e+07
cache size (bytes)



opt3 size 1000
time- 714



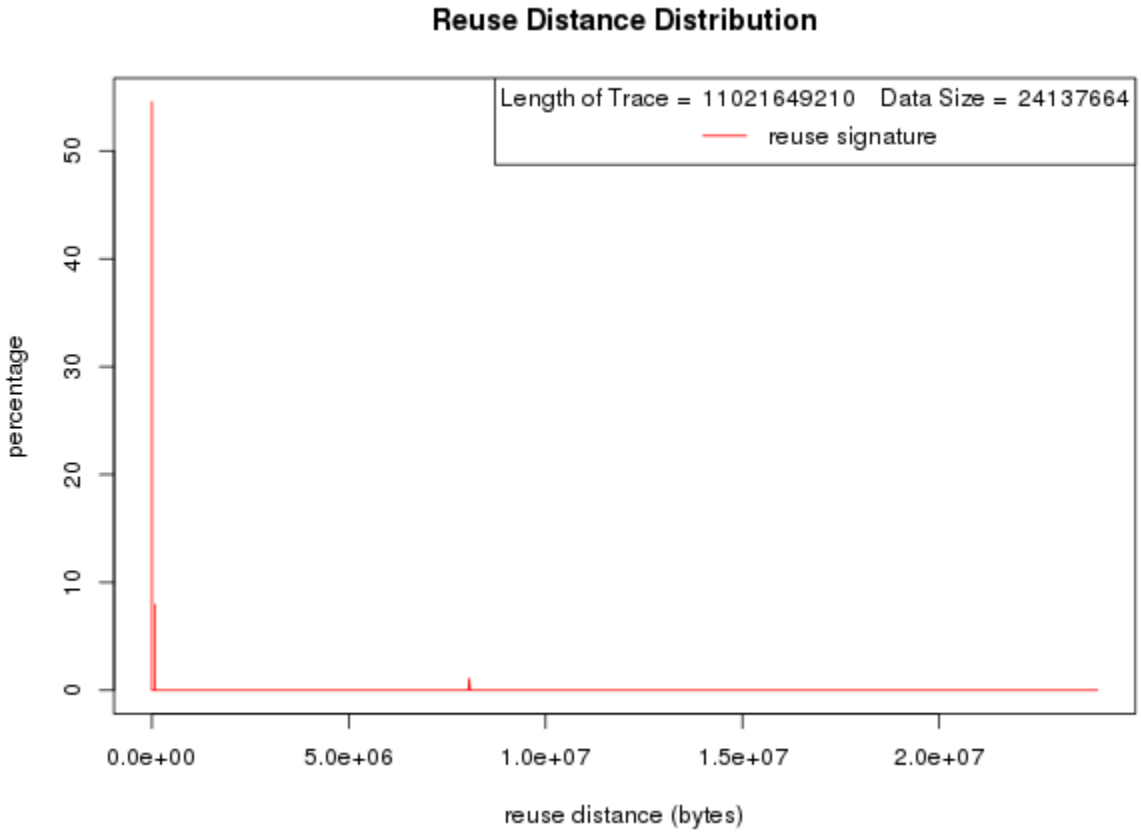


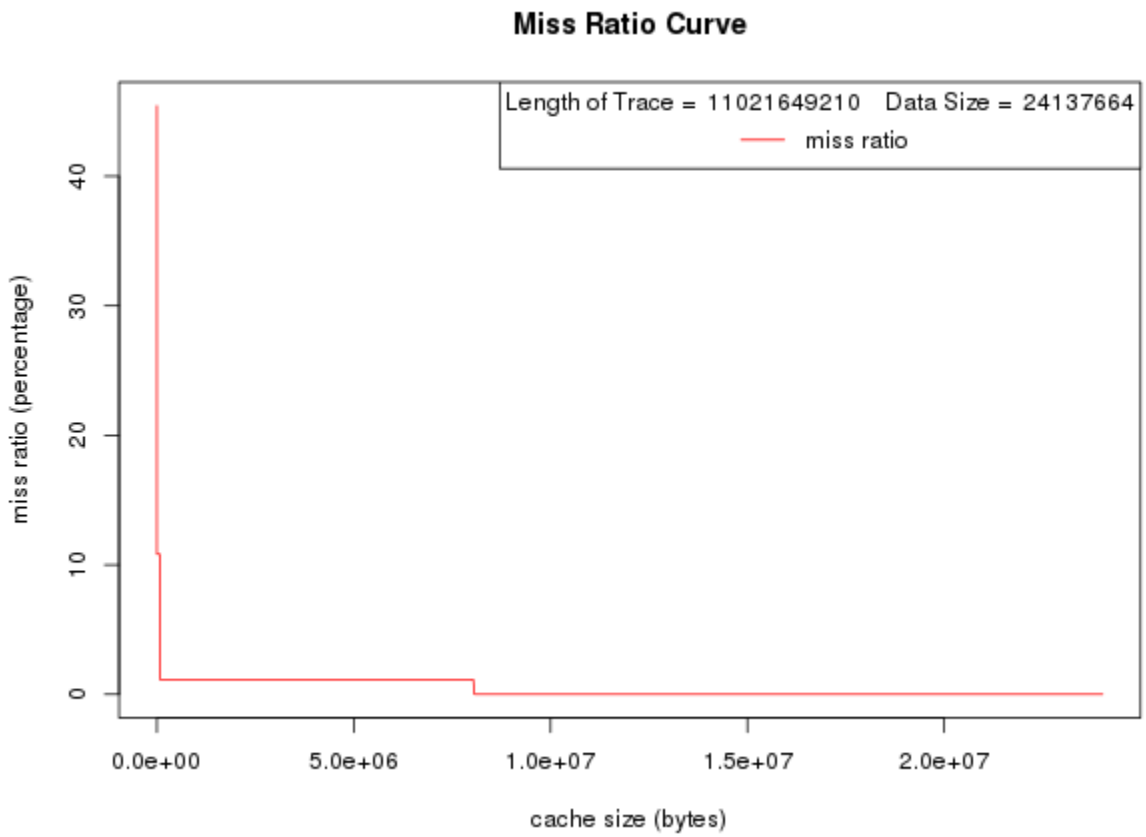
Several conclusions can be drawn from the data above. The average footprint for small matrices seems in increase in size in a somewhat linear fashion with time. For large matricies, in turn show a similar relationship. Their footprint increases at an extremely high right for the first few hundred calculation, and then adopts a (by comparison) extremely shallow, linear increase in footprint size.

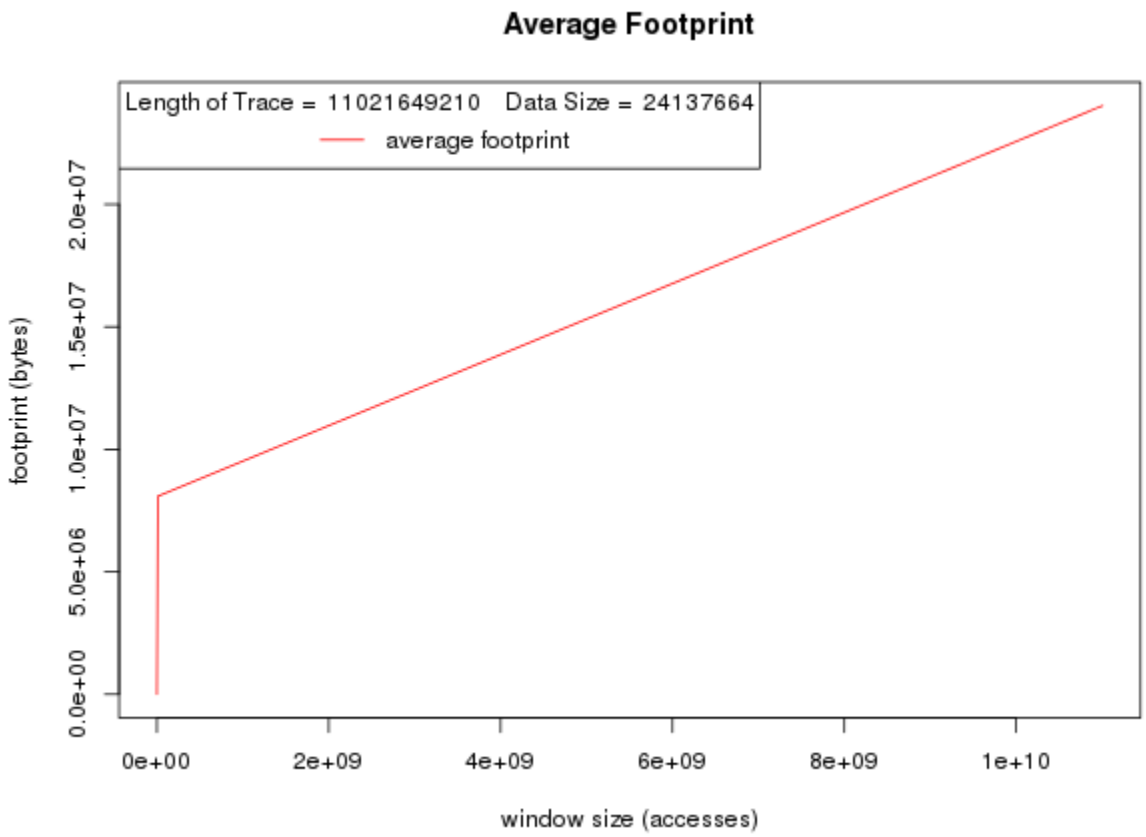
Reuse distance shows almost identical behavior in both small and large matrices. In both cases, reuse distance decreases by over 70 percent after the first few calculations. A similar phenomena is noted in the miss ratio measurements, which drop drastically after the few moments of the program. In contrast to the reuse distance, however, the miss ratio makes a second drop (in a step-wise manner) at

approximately 5.5*10^6 milliseconds.

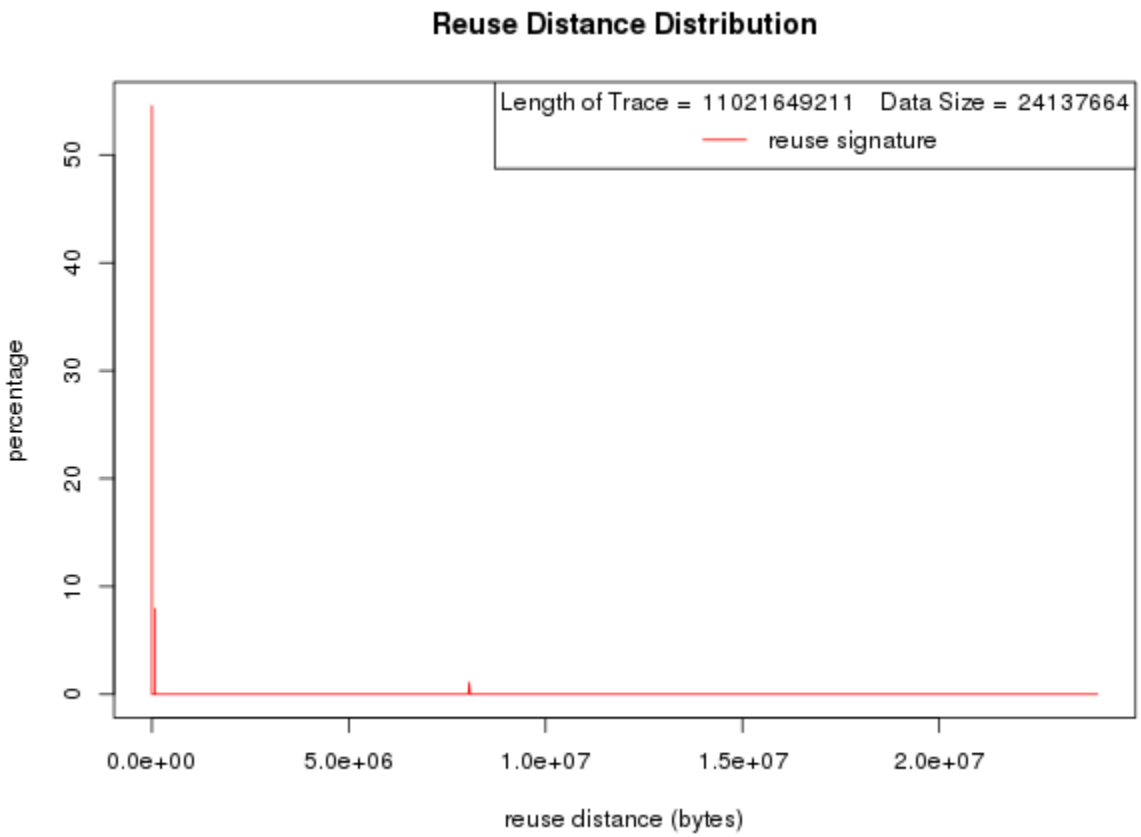
The run time graphs demonstrate a peculiar relationship between compiler optimization and the time needed to execute a program. It would seem that each additional level of optimization creates an increase in run time (with a more pronounced effect on the programs computing large matrices), up until the third optimization flag is used. At this point, it actually decreases the run time.

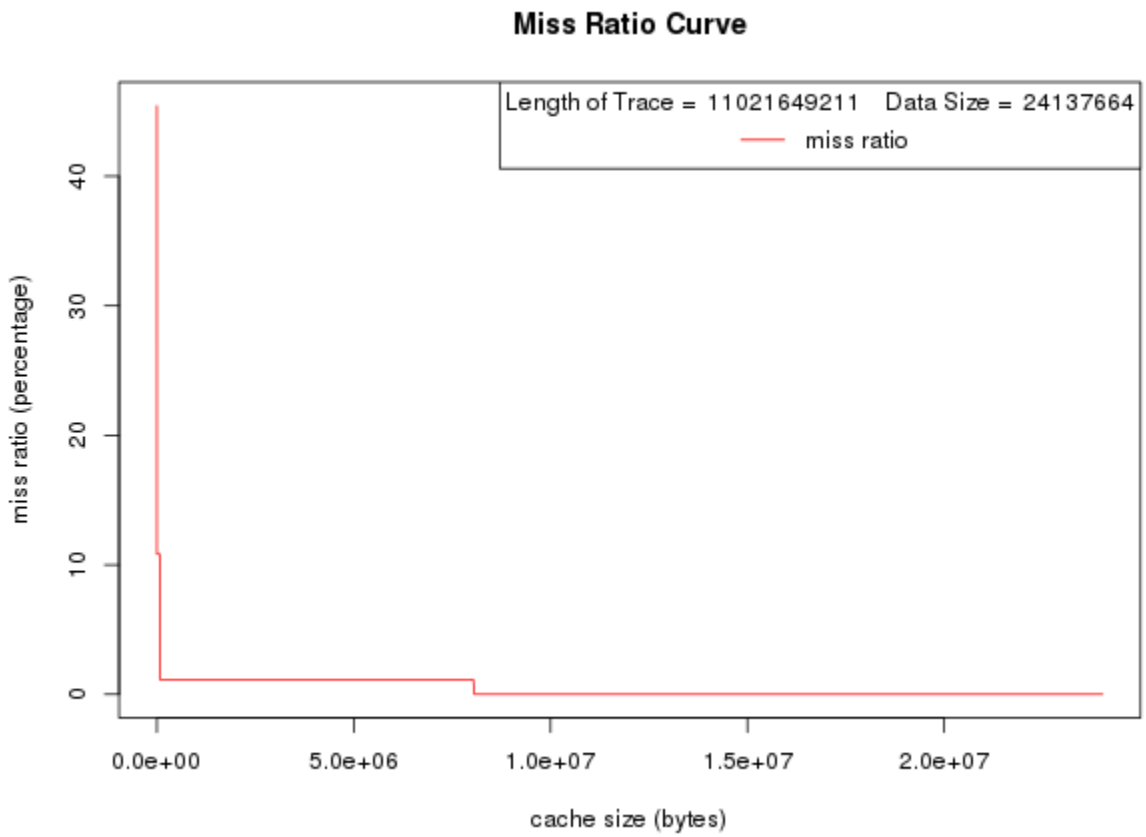


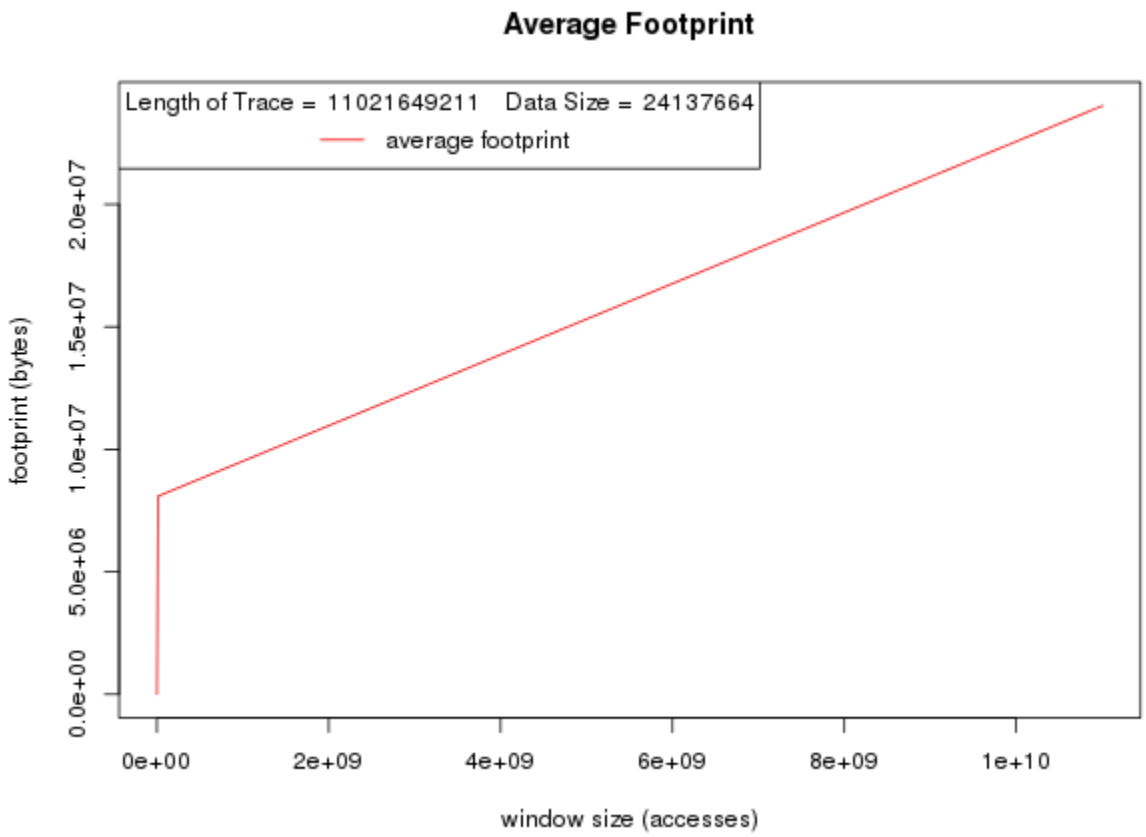

Miss Ratio Curve
Length of Trace = 11021649210 Data Size = 24137664
miss ratio
miss ratio (percentage)
0
10
20
30
40
0.0e+00
5.0e+06
1.0e+07
1.5e+07
2.0e+07
cache size (bytes)



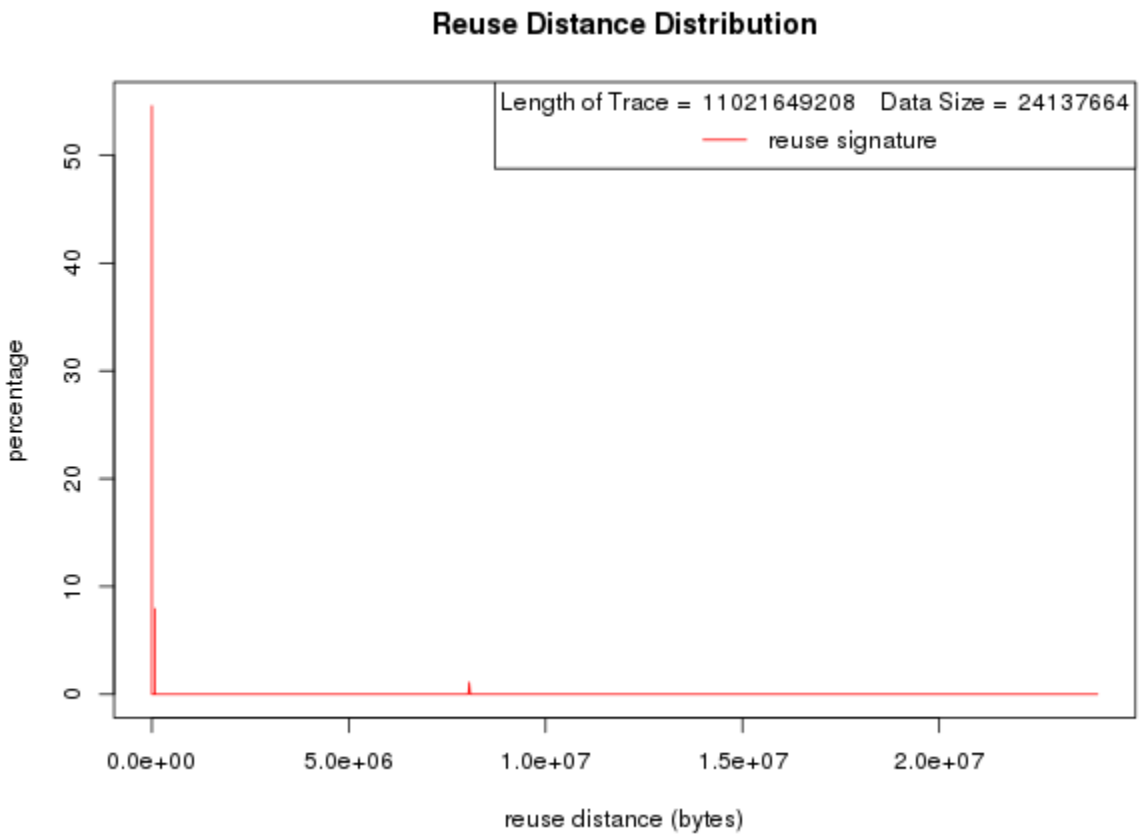
tiled- 5
runtime- 902.94

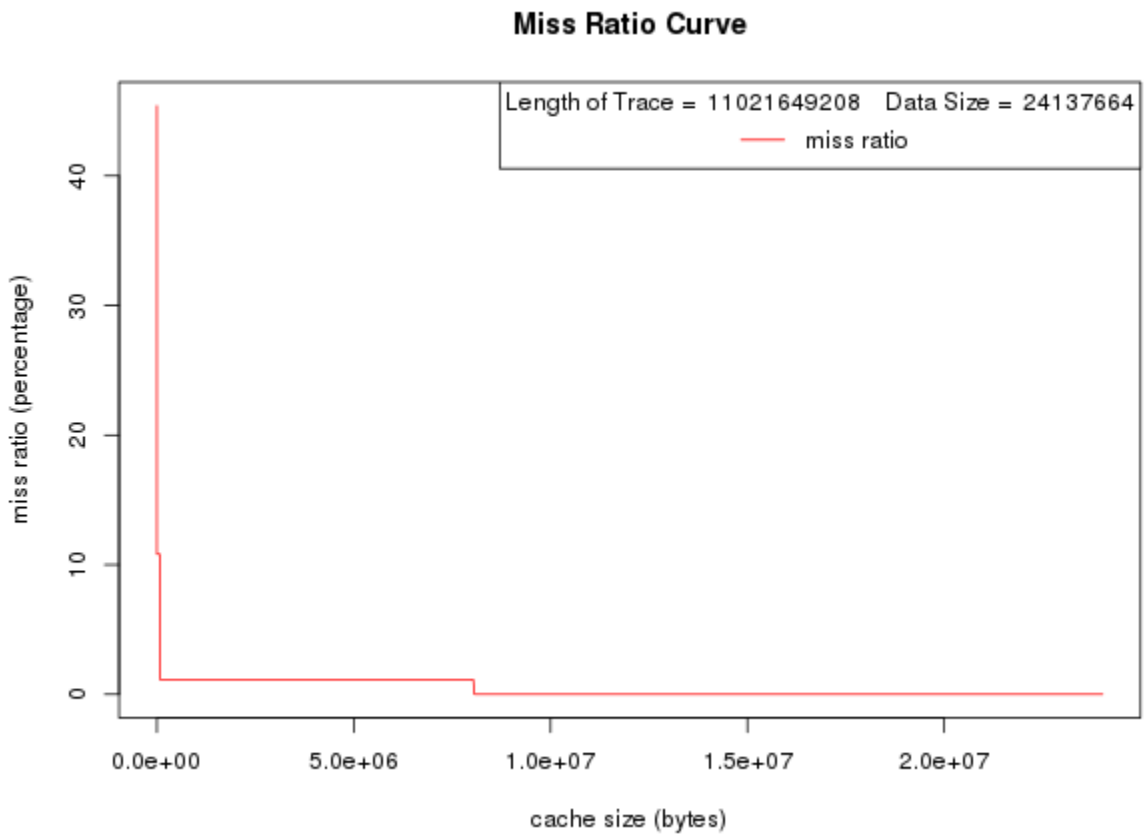

Reuse Distance Distribution
Length of Trace = 11021649211 Data Size = 24137664
reuse signature
percentage
0
10
20
30
40
50
0.0e+00
5.0e+06
1.0e+07
1.5e+07
2.0e+07
reuse distance (bytes)

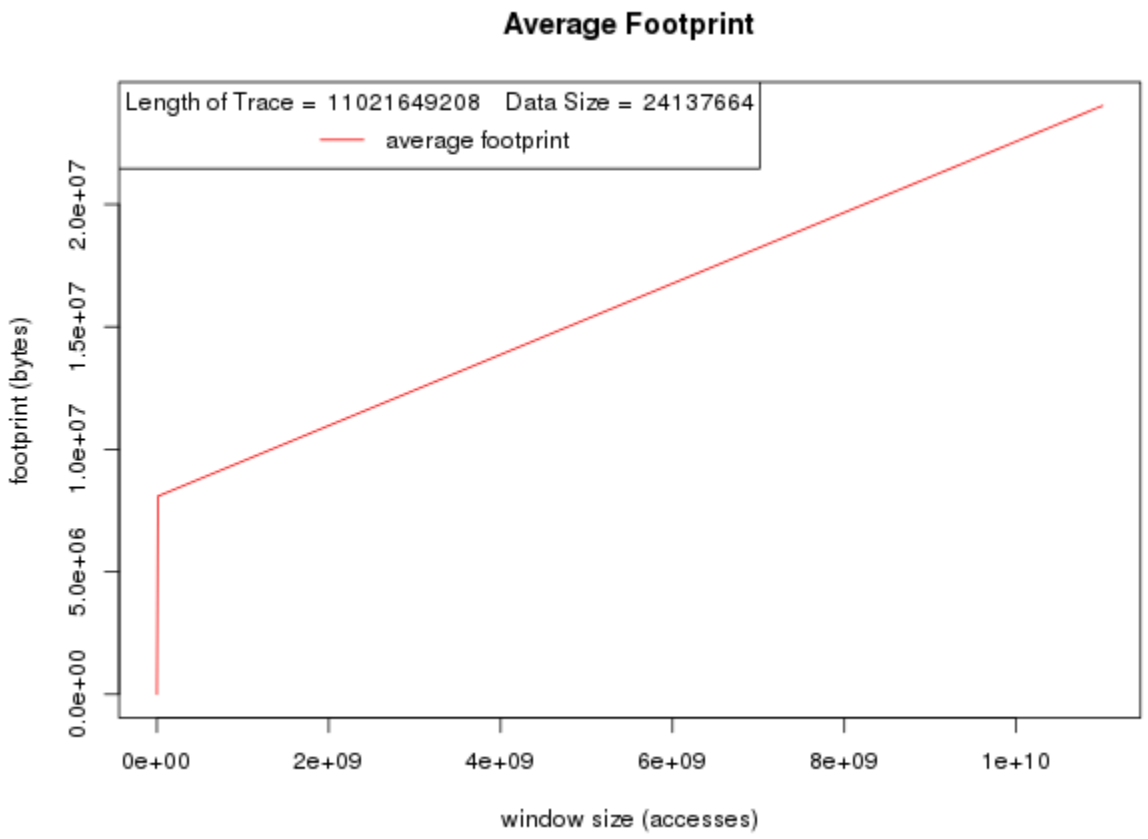

Miss Ratio Curve
Length of Trace = 11021649211 Data Size = 24137664
miss ratio
miss ratio (percentage)
0
10
20
30
40
0.0e+00
5.0e+06
1.0e+07
1.5e+07
2.0e+07
cache size (bytes)



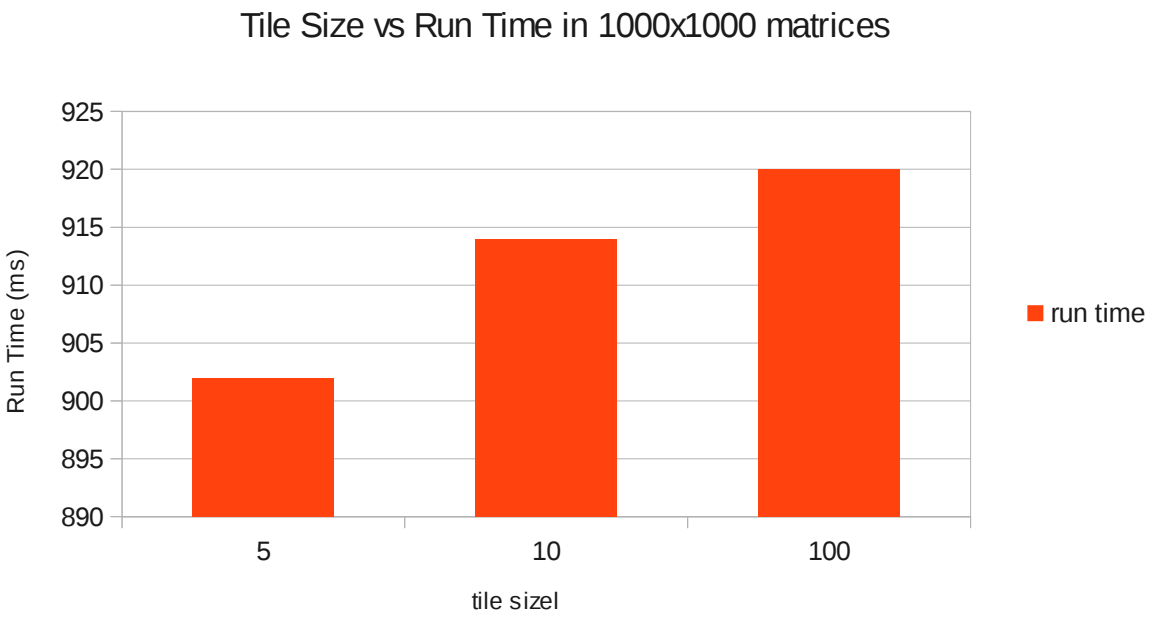
tiled- 10
time- 914.74


Reuse Distance Distribution
Length of Trace = 11021649208 Data Size = 24137664
reuse signature
percentage
0
10
20
30
40
50
0.0e+00
5.0e+06
1.0e+07
1.5e+07
2.0e+07
reuse distance (bytes)


Miss Ratio Curve
Length of Trace = 11021649208 Data Size = 24137664
miss ratio
miss ratio (percentage)
0
10
20
30
40
0.0e+00
5.0e+06
1.0e+07
1.5e+07
2.0e+07
cache size (bytes)



tiled- 100
runtime- 920.5



From this graph, we can conclude that tiling does not appear to decrease the run time of the program for large matrices with the third optimization flag. In fact, it seems that increased tiling only increases run time. Even further, the best time for the tiled run (902 ms) is almost 200 ms longer than the longest run time of the non-tiled version. This indicates quite clearly that tiling offers no improvement to a problem of these particular parameters, and further, actually is detrimental.

In terms of locality, tiling did not offer any clear advantage of non-tiling programs. The patterns and values for all three measures of locality were nearly identical to those measurements without tiling. From all of the following, we can conclude that tiling offers no measurable improvement over non-tiled programs.