Profile Report

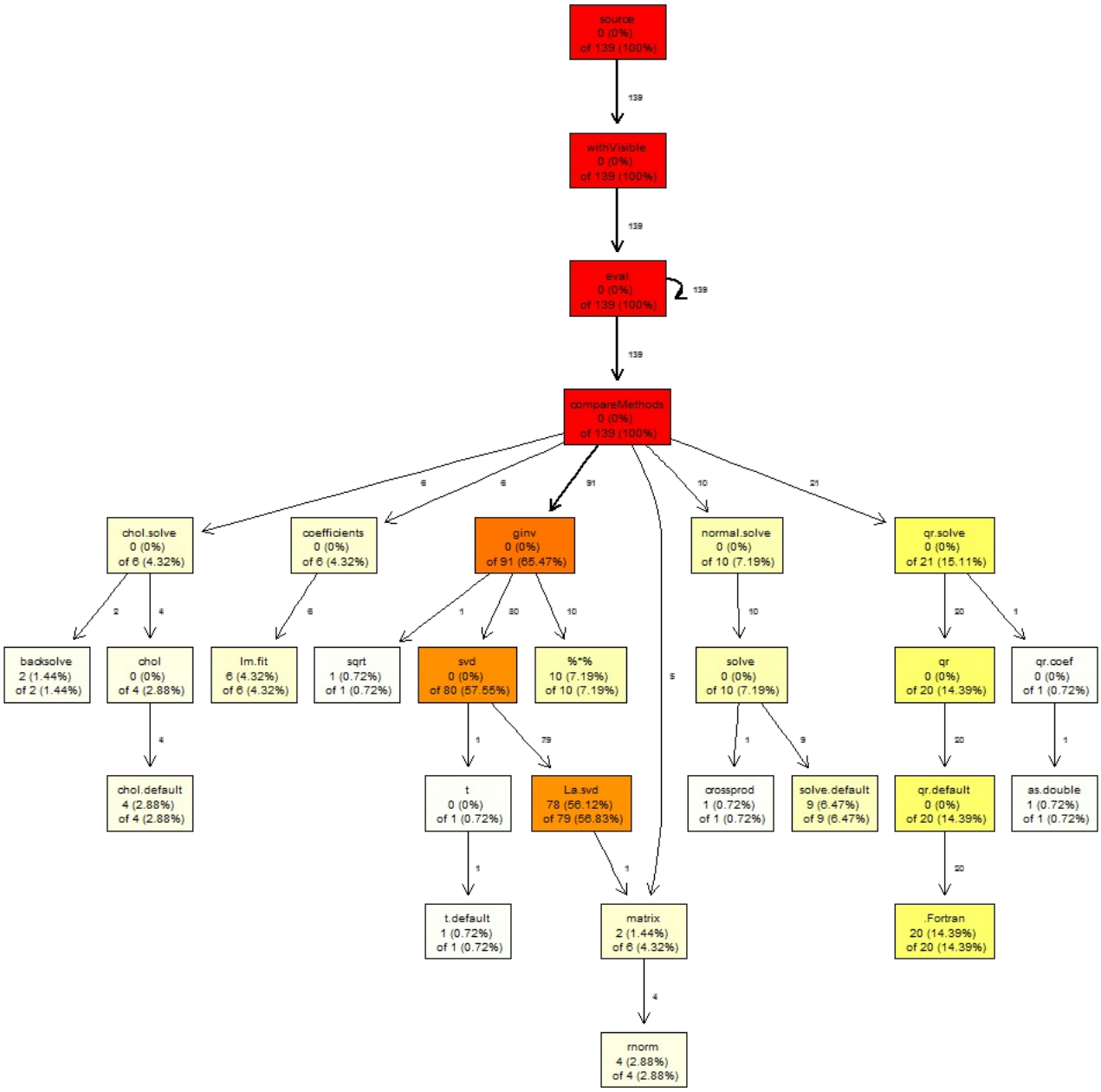
Summary

Table 1.

File	Total time	Self time	Total time (%)	Self time (%)
compareMethods	2.8	2.5	100	88
2016-09-08	2.8	0	100	0
normal.solve	0.2	0.2	7.2	7.2
chol.solve	0.12	0.12	4.3	4.3

Call graph

Figure 1.



compareMethods

time	line	
	1	compareMethods <-

```
      2  function() {
      3  library(MASS)
      4  # Call the functions
      5  source(paste(temp,"/normal.solve.R",sep=""))
      6  source(paste(temp,"/chol.solve.R",sep=""))
      7  # Solving a big system of equations
      8  nrows <- 1000
      9  ncols <- 500
0.1  10  A <- matrix(rnorm(nrows*ncols),nrows,ncols)
     11  b <- rnorm(nrows)
     12  # Testing different possibilities
0.42 13  Sol1 <- qr.solve(A,b) # Using QR factorization
0.12 14  Sol2 <- coefficients(lm.fit(A,b)) # lm.fit, based on QR but with some overhead
1.82 15  Sol3 <- ginv(A) %*% b # Using the pseudoinverse based on SVD
0.2  16  Sol4 <- normal.solve(A,b) # Using a function based on the normal equations.
0.12 17  Sol5 <- chol.solve(A,b) # Using a function based on the Choleski factorization.
     18  }
```

2016-09-08

time	line	
	1	library('GUIProfiler')
	2	
	3	temp<-tempdir()
	4	# Definition of two functions
	5	normal.solve <- function(A,b) {
	6	Output <- solve(crossprod(A), t(A)%*%b)
	7	}
	8	
	9	chol.solve <- function(A,b) {
	10	L <- chol(crossprod(A))
	11	Output1 <- backsolve(L, t(A)%*%b, transpose=TRUE)
	12	Output2 <- backsolve(L, Output1)
	13	}
	14	
	15	compareMethods <- function() {
	16	library(MASS)
	17	# Call the functions
	18	source(paste(temp,"/normal.solve.R",sep=""))
	19	source(paste(temp,"/chol.solve.R",sep=""))
	20	# Solving a big system of equations
	21	nrows <- 1000
	22	ncols <- 500
	23	A <- matrix(rnorm(nrows*ncols),nrows,ncols)
	24	b <- rnorm(nrows)
	25	# Testing different possibilities
	26	Sol1 <- qr.solve(A,b) # Using QR factorization
	27	Sol2 <- coefficients(lm.fit(A,b)) # lm.fit, based on QR but with some overhead
	28	Sol3 <- ginv(A) %*% b # Using the pseudoinverse based on SVD
	29	Sol4 <- normal.solve(A,b) # Using a function based on the normal equations.
	30	Sol5 <- chol.solve(A,b) # Using a function based on the Choleski factorization.
	31	}
	32	
	33	# Dump these functions to three different files
	34	
	35	dump("normal.solve",file=paste(temp,"/normal.solve.R",sep=""))
	36	dump("chol.solve",file=paste(temp,"/chol.solve.R",sep=""))
	37	dump("compareMethods",file=paste(temp,"/compareMethods.R",sep=""))
	38	source(paste(temp,"/compareMethods.R",sep=""))
	39	
	40	# Profile the code
	41	
	42	RRprofStart()
2.78	43	compareMethods()
	44	RRprofStop()
	45	# Uncomment to open the report
```
```

```
46 #RRprofReport()
47
48 RRprofReport(file.name = "RRprof.out", notepad.path =
49 "d:/Program Files (x86)/Notepad++/notepad++.exe",reportname = "my_report")
```

normal.solve

time	line	
	1	normal.solve <-
	2	function(A,b) {
0.2	3	Output <- solve(crossprod(A), t(A)%*%b)
	4	}

chol.solve

time	line	
	1	chol.solve <-
	2	function(A,b) {
0.08	3	L <- chol(crossprod(A))
0.04	4	Output1 <- backsolve(L, t(A)%*%b, transpose=TRUE)
	5	Output2 <- backsolve(L, Output1)
	6	}