

```

1  Assume R has been built with debug=T
2  brb@brb-T3500:~/Downloads/R-3.2.2$ ./configure
3  brb@brb-T3500:~/Downloads/R-3.2.2$ make debug=T
4
5  brb@brb-T3500:~/Downloads/R-3.2.2$ bin/R -d gdb --vanilla
6  GNU gdb (Ubuntu 7.7.1-0ubuntu5~14.04.2) 7.7.1
7  Copyright (C) 2014 Free Software Foundation, Inc.
8  License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
9  This is free software: you are free to change and redistribute it.
10 There is NO WARRANTY, to the extent permitted by law.  Type "show copying"
11 and "show warranty" for details.
12 This GDB was configured as "x86_64-linux-gnu".
13 Type "show configuration" for configuration details.
14 For bug reporting instructions, please see:
15 <http://www.gnu.org/software/gdb/bugs/>.
16 Find the GDB manual and other documentation resources online at:
17 <http://www.gnu.org/software/gdb/documentation/>.
18 For help, type "help".
19 Type "apropos word" to search for commands related to "word"...
20 Reading symbols from /home/brb/Downloads/R-3.2.2/bin/exec/R...done.
21 (gdb) run
22 Starting program: /home/brb/Downloads/R-3.2.2/bin/exec/R --vanilla
23 [Thread debugging using libthread_db enabled]
24 Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".
25
26 R version 3.2.2 (2015-08-14) -- "Fire Safety"
27 Copyright (C) 2015 The R Foundation for Statistical Computing
28 Platform: x86_64-pc-linux-gnu (64-bit)
29
30 R is free software and comes with ABSOLUTELY NO WARRANTY.
31 You are welcome to redistribute it under certain conditions.
32 Type 'license()' or 'licence()' for distribution details.
33
34   Natural language support but running in an English locale
35
36 R is a collaborative project with many contributors.
37 Type 'contributors()' for more information and
38 'citation()' on how to cite R or R packages in publications.
39
40 Type 'demo()' for some demos, 'help()' for on-line help, or
41 'help.start()' for an HTML browser interface to help.
42 Type 'q()' to quit R.
43
44 > debug(cor)
45 > cor(1:5, rnorm(5))
46 debugging in: cor(1:5, rnorm(5))
47 debug: {
48     na.method <- pmatch(use, c("all.obs", "complete.obs", "pairwise.complete.obs",
49     "everything", "na.or.complete"))
50     if (is.na(na.method))
51         stop("invalid 'use' argument")
52     method <- match.arg(method)
53     if (is.data.frame(y))
54         y <- as.matrix(y)
55     if (is.data.frame(x))
56         x <- as.matrix(x)
57     if (!is.matrix(x) && is.null(y))

```

```

58     stop("supply both 'x' and 'y' or a matrix-like 'x'")
59 if (!(is.numeric(x) || is.logical(x)))
60     stop("'x' must be numeric")
61 stopifnot(is.atomic(x))
62 if (!is.null(y)) {
63     if (!(is.numeric(y) || is.logical(y)))
64         stop("'y' must be numeric")
65     stopifnot(is.atomic(y))
66 }
67 Rank <- function(u) {
68     if (length(u) == 0L)
69         u
70     else if (is.matrix(u)) {
71         if (nrow(u) > 1L)
72             apply(u, 2L, rank, na.last = "keep")
73         else row(u)
74     }
75     else rank(u, na.last = "keep")
76 }
77 if (method == "pearson")
78     .Call(C_cor, x, y, na.method, FALSE)
79 else if (na.method %in% c(2L, 5L)) {
80     if (is.null(y)) {
81         .Call(C_cor, Rank(na.omit(x)), NULL, na.method, method ==
82             "kendall")
83     }
84     else {
85         nas <- attr(na.omit(cbind(x, y)), "na.action")
86         dropNA <- function(x, nas) {
87             if (length(nas)) {
88                 if (is.matrix(x))
89                     x[-nas, , drop = FALSE]
90                 else x[-nas]
91             }
92             else x
93         }
94         .Call(C_cor, Rank(dropNA(x, nas)), Rank(dropNA(y,
95             nas)), na.method, method == "kendall")
96     }
97 }
98 else if (na.method != 3L) {
99     x <- Rank(x)
100     if (!is.null(y))
101         y <- Rank(y)
102     .Call(C_cor, x, y, na.method, method == "kendall")
103 }
104 else {
105     if (is.null(y)) {
106         ncy <- ncx <- ncol(x)
107         if (ncx == 0)
108             stop("'x' is empty")
109         r <- matrix(0, nrow = ncx, ncol = ncy)
110         for (i in seq_len(ncx)) {
111             for (j in seq_len(i)) {
112                 x2 <- x[, i]
113                 y2 <- x[, j]
114                 ok <- complete.cases(x2, y2)

```

```

115         x2 <- rank(x2[ok])
116         y2 <- rank(y2[ok])
117         r[i, j] <- if (any(ok))
118             .Call(C_cor, x2, y2, 1L, method == "kendall")
119         else NA
120     }
121 }
122 r <- r + t(r) - diag(diag(r))
123 rownames(r) <- colnames(x)
124 colnames(r) <- colnames(x)
125 r
126 }
127 else {
128     if (length(x) == 0L || length(y) == 0L)
129         stop("both 'x' and 'y' must be non-empty")
130     matrix_result <- is.matrix(x) || is.matrix(y)
131     if (!is.matrix(x))
132         x <- matrix(x, ncol = 1L)
133     if (!is.matrix(y))
134         y <- matrix(y, ncol = 1L)
135     ncx <- ncol(x)
136     ncy <- ncol(y)
137     r <- matrix(0, nrow = ncx, ncol = ncy)
138     for (i in seq_len(ncx)) {
139         for (j in seq_len(ncy)) {
140             x2 <- x[, i]
141             y2 <- y[, j]
142             ok <- complete.cases(x2, y2)
143             x2 <- rank(x2[ok])
144             y2 <- rank(y2[ok])
145             r[i, j] <- if (any(ok))
146                 .Call(C_cor, x2, y2, 1L, method == "kendall")
147             else NA
148         }
149     }
150     rownames(r) <- colnames(x)
151     colnames(r) <- colnames(y)
152     if (matrix_result)
153         r
154     else drop(r)
155 }
156 }
157 }
158 Browse[2]>
159 debug: na.method <- pmatch(use, c("all.obs", "complete.obs", "pairwise.complete.obs",
160     "everything", "na.or.complete"))
161 Browse[2]>
162 debug: if (is.na(na.method)) stop("invalid 'use' argument")
163 Browse[2]>
164 debug: method <- match.arg(method)
165 Browse[2]>
166 debug: if (is.data.frame(y)) y <- as.matrix(y)
167 Browse[2]>
168 debug: if (is.data.frame(x)) x <- as.matrix(x)
169 Browse[2]>
170 debug: if (!is.matrix(x) && is.null(y)) stop("supply both 'x' and 'y' or a matrix-like 'x'")
171 Browse[2]>

```

```

172 debug: if (!(is.numeric(x) || is.logical(x))) stop("'x' must be numeric")
173 Browse[2]>
174 debug: stopifnot(is.atomic(x))
175 Browse[2]>
176 debug: if (!is.null(y)) {
177     if (!(is.numeric(y) || is.logical(y)))
178         stop("'y' must be numeric")
179     stopifnot(is.atomic(y))
180 }
181 Browse[2]>
182 debug: if (!(is.numeric(y) || is.logical(y))) stop("'y' must be numeric")
183 Browse[2]>
184 debug: stopifnot(is.atomic(y))
185 Browse[2]>
186 debug: Rank <- function(u) {
187     if (length(u) == 0L)
188         u
189     else if (is.matrix(u)) {
190         if (nrow(u) > 1L)
191             apply(u, 2L, rank, na.last = "keep")
192         else row(u)
193     }
194     else rank(u, na.last = "keep")
195 }
196 Browse[2]>
197 debug: if (method == "pearson") .Call(C_cor, x, y, na.method, FALSE) else if (na.method %in%
198     c(2L, 5L)) {
199     if (is.null(y)) {
200         .Call(C_cor, Rank(na.omit(x)), NULL, na.method, method ==
201             "kendall")
202     }
203     else {
204         nas <- attr(na.omit(cbind(x, y)), "na.action")
205         dropNA <- function(x, nas) {
206             if (length(nas)) {
207                 if (is.matrix(x))
208                     x[-nas, , drop = FALSE]
209                 else x[-nas]
210             }
211             else x
212         }
213         .Call(C_cor, Rank(dropNA(x, nas)), Rank(dropNA(y, nas)),
214             na.method, method == "kendall")
215     }
216 } else if (na.method != 3L) {
217     x <- Rank(x)
218     if (!is.null(y))
219         y <- Rank(y)
220     .Call(C_cor, x, y, na.method, method == "kendall")
221 } else {
222     if (is.null(y)) {
223         ncy <- ncx <- ncol(x)
224         if (ncx == 0)
225             stop("'x' is empty")
226         r <- matrix(0, nrow = ncx, ncol = ncy)
227         for (i in seq_len(ncx)) {
228             for (j in seq_len(i)) {

```

```

229         x2 <- x[, i]
230         y2 <- x[, j]
231         ok <- complete.cases(x2, y2)
232         x2 <- rank(x2[ok])
233         y2 <- rank(y2[ok])
234         r[i, j] <- if (any(ok))
235             .Call(C_cor, x2, y2, 1L, method == "kendall")
236         else NA
237     }
238 }
239 r <- r + t(r) - diag(diag(r))
240 rownames(r) <- colnames(x)
241 colnames(r) <- colnames(x)
242 r
243 }
244 else {
245     if (length(x) == 0L || length(y) == 0L)
246         stop("both 'x' and 'y' must be non-empty")
247     matrix_result <- is.matrix(x) || is.matrix(y)
248     if (!is.matrix(x))
249         x <- matrix(x, ncol = 1L)
250     if (!is.matrix(y))
251         y <- matrix(y, ncol = 1L)
252     ncx <- ncol(x)
253     ncy <- ncol(y)
254     r <- matrix(0, nrow = ncx, ncol = ncy)
255     for (i in seq_len(ncx)) {
256         for (j in seq_len(ncy)) {
257             x2 <- x[, i]
258             y2 <- y[, j]
259             ok <- complete.cases(x2, y2)
260             x2 <- rank(x2[ok])
261             y2 <- rank(y2[ok])
262             r[i, j] <- if (any(ok))
263                 .Call(C_cor, x2, y2, 1L, method == "kendall")
264             else NA
265         }
266     }
267     rownames(r) <- colnames(x)
268     colnames(r) <- colnames(y)
269     if (matrix_result)
270         r
271     else drop(r)
272 }
273 }
274 Browse[2]>
275 debug: .Call(C_cor, x, y, na.method, FALSE)
276 Browse[2]> # Press Ctrl + c
277 Program received signal SIGINT, Interrupt.
278 0x00007ffff62cbd83 in __select_nocancel ()
279     at ../sysdeps/unix/syscall-template.S:81
280 81     ../sysdeps/unix/syscall-template.S: No such file or directory.
281 (gdb)
282 (gdb) b corcov
283 Breakpoint 1 at 0x7ffff46167c0: file cov.c, line 637.
284 (gdb) c
285 Continuing.

```

```

286 # Press Return key
287
288 Breakpoint 1, corcov (x=0x110b348, y=0x153bf00, na_method=0xb17d18,
289     skendall=0xc88788, cor=TRUE) at cov.c:637
290 637 {
291 (gdb) backtrace
292 #0 corcov (x=<optimized out>, y=0x153bf00, na_method=<optimized out>,
293     skendall=<optimized out>, cor=TRUE) at cov.c:741
294 #1 0x0000000000480120 in do_dotcall (call=0xe36030, op=<optimized out>,
295     args=<optimized out>, env=<optimized out>) at dotcode.c:1251
296 #2 0x00000000004bd03f in Rf_eval (e=0xe36030, rho=0xcefcf0) at eval.c:655
297 #3 0x00000000004bcea9 in Rf_eval (e=0xe36c18, rho=rho@entry=0xcefcf0)
298     at eval.c:627
299 #4 0x00000000004befd9 in do_begin (call=0xe3eb58, op=0x97c678,
300     args=0xe36be0, rho=0xcefcf0) at eval.c:1716
301 #5 0x00000000004bcea9 in Rf_eval (e=0xe3eb58, rho=0xcefcf0) at eval.c:627
302 #6 0x00000000004be0ff in Rf_applyClosure (call=call@entry=0xcf2df0,
303     op=op@entry=0xe3f668, arglist=0xcefa50, rho=rho@entry=0x9a5e68,
304     suppliedvars=0x96f928) at eval.c:1039
305 #7 0x00000000004bccd9 in Rf_eval (e=e@entry=0xcf2df0, rho=rho@entry=0x9a5e68)
306     at eval.c:674
307 #8 0x00000000004e2041 in Rf_ReplIteration (rho=rho@entry=0x9a5e68,
308     savestack=savestack@entry=0, browselevel=browselevel@entry=0,
309     state=state@entry=0x7fffffffcd0) at main.c:258
310 #9 0x00000000004e2368 in R_ReplConsole (rho=0x9a5e68, savestack=0,
311     browselevel=0) at main.c:308
312 #10 0x00000000004e2411 in run_Rmainloop () at main.c:1006
313 #11 0x00000000004e2452 in Rf_mainloop () at main.c:1013
314 #12 0x0000000000418b78 in main (ac=ac@entry=2, av=av@entry=0x7fffffffdef8)
315     ---Type <return> to continue, or q <return> to quit---
316     at Rmain.c:29
317 #13 0x00007ffff61fbec5 in __libc_start_main (main=0x418b60 <main>, argc=2,
318     argv=0x7fffffffdef8, init=<optimized out>, fini=<optimized out>,
319     rtdl_fini=<optimized out>, stack_end=0x7fffffffdee8) at libc-start.c:287
320 #14 0x0000000000418ba8 in _start ()
321
322 (gdb) next
323 643     if(isNull(x)) /* never allowed */
324     .....
325 771     if (ansmat) { /* set dimnames() when applicable */
326 (gdb)
327 797     if(sd_0)/* only in cor() */
328 (gdb)
329 799     UNPROTECT(nprotect);
330 (gdb)
331 801 }
332 (gdb)
333 do_dotcall (call=0xe36030, op=<optimized out>, args=<optimized out>,
334     env=<optimized out>) at dotcode.c:1252
335 1252     vmaxset(vmax);
336 (gdb)
337 1251     retval = R_doDotCall(ofun, nargs, cargs, call);
338 (gdb)
339 1252     vmaxset(vmax);
340 (gdb)
341 1254 }
342 (gdb)

```

```

343 Rf_eval (e=0xe36030, rho=0xcefcf0) at eval.c:656
344 656 R_Srcref = oldref;
345 (gdb)
346 657 endcontext(&cntxt);
347 (gdb)
348 656 R_Srcref = oldref;
349 ...
350 (gdb)
351 Rf_eval (e=0xe36c18, rho=rho@entry=0xcefcf0) at eval.c:637
352 637 if (flag < 2) R_Visible = flag != 1;
353 (gdb)
354 669 check_stack_balance(op, save);
355 ...
356 do_begin (call=0xe3eb58, op=0x97c678, args=0xe36be0, rho=0xcefcf0)
357 at eval.c:1709
358 1709 while (args != R_NilValue) {
359 (gdb)
360 1718 args = CDR(args);
361 ....
362 Rf_eval (e=0xe3eb58, rho=0xcefcf0) at eval.c:637
363 637 if (flag < 2) R_Visible = flag != 1;
364 (gdb)
365 669 check_stack_balance(op, save);
366 ...
367 (gdb)
368 Rf_applyClosure (call=call@entry=0xcf2df0, op=op@entry=0xe3f668,
369 arglist=0xcefa50, rho=rho@entry=0x9a5e68, suppliedvars=0x96f928)
370 at eval.c:1042
371 1042 endcontext(&cntxt);
372 ...
373 1045 Rprintf("exiting from: ");
374 (gdb)
375 exiting from: 1046 PrintValueRec(call, rho);
376 (gdb)
377 cor(1:5, rnorm(5))
378 1048 UNPROTECT(3);
379 ...
380 Rf_eval (e=e@entry=0xcf2df0, rho=rho@entry=0x9a5e68) at eval.c:675
381 675 UNPROTECT(1);
382 (gdb)
383 ...
384 Rf_eval (e=e@entry=0xcf2df0, rho=rho@entry=0x9a5e68) at eval.c:675
385 675 UNPROTECT(1);
386 (gdb)
387 ...
388 Rf_ReplIteration (rho=rho@entry=0x9a5e68, savestack=savestack@entry=0,
389 browselevel=browselevel@entry=0, state=state@entry=0x7fffffffcdc0)
390 at main.c:259
391 259 SET_SYMVALUE(R_LastvalueSymbol, value);
392 260 wasDisplayed = R_Visible;
393 (gdb)
394 261 if (R_Visible)
395 (gdb)
396 262 PrintValueEnv(value, rho);
397 (gdb)
398 [1] -0.6156519
399 ...

```

```
400 R_ReplConsole (rho=0x9a5e68, savestack=0, browselevel=0) at main.c:309
401 309         if(status < 0) {
402 (gdb)
403 308         status = Rf_ReplIteration(rho, savestack, browselevel, &state);
404 (gdb)
405 >
406 309         if(status < 0) {
407 (gdb)
408 308         status = Rf_ReplIteration(rho, savestack, browselevel, &state);
409 (gdb)
410 >
411 [Inf loop]
412 > q('no')
413 [Inferior 1 (process 25390) exited normally]
414 (gdb) quit
415 brb@brb-T3500:~/Downloads/R-3.2.2$
416
```