Assume R has been built with debug=T

```
brb@brb-T3500:~/Downloads/R-3.2.2$ ./configure
brb@brb-T3500:~/Downloads/R-3.2.2$ make debug=T

brb@brb-T3500:~/Downloads/R-3.2.2$ bin/R -d gdb --vanilla
GNU gdb (Ubuntu 7.7.1-0ubuntu5~14.04.2) 7.7.1
Copyright (C) 2014 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.  Type "show copying"
and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
<http://www.gnu.org/software/gdb/documentation/>.
For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from /home/brb/Downloads/R-3.2.2/bin/exec/R...done.
(gdb) run
Starting program: /home/brb/Downloads/R-3.2.2/bin/exec/R --vanilla
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".

R version 3.2.2 (2015-08-14) -- "Fire Safety"
Copyright (C) 2015 The R Foundation for Statistical Computing
Platform: x86_64-pc-linux-gnu (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

  Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

> debug(cor)
> cor(1:5, rnorm(5))
debugging in: cor(1:5, rnorm(5))
debug: {
    na.method <- pmatch(use, c("all.obs", "complete.obs", "pairwise.complete.obs",
        "everything", "na.or.complete"))
    if (is.na(na.method))
        stop("invalid 'use' argument")
    method <- match.arg(method)
    if (is.data.frame(y))
        y <- as.matrix(y)
    if (is.data.frame(x))
        x <- as.matrix(x)
    if (!is.matrix(x) && is.null(y))
```

```
            stop("supply both 'x' and 'y' or a matrix-like 'x'")
        if (!(is.numeric(x) || is.logical(x)))
            stop("'x' must be numeric")
        stopifnot(is.atomic(x))
        if (!is.null(y)) {
            if (!(is.numeric(y) || is.logical(y)))
                stop("'y' must be numeric")
            stopifnot(is.atomic(y))
        }
        Rank <- function(u) {
            if (length(u) == 0L)
                u
            else if (is.matrix(u)) {
                if (nrow(u) > 1L)
                    apply(u, 2L, rank, na.last = "keep")
                else row(u)
            }
            else rank(u, na.last = "keep")
        }
        if (method == "pearson")
            .Call(C_cor, x, y, na.method, FALSE)
        else if (na.method %in% c(2L, 5L)) {
            if (is.null(y)) {
                .Call(C_cor, Rank(na.omit(x)), NULL, na.method, method ==
                    "kendall")
            }
            else {
                nas <- attr(na.omit(cbind(x, y)), "na.action")
                dropNA <- function(x, nas) {
                    if (length(nas)) {
                      if (is.matrix(x))
                        x[-nas, , drop = FALSE]
                      else x[-nas]
                    }
                    else x
                }
                .Call(C_cor, Rank(dropNA(x, nas)), Rank(dropNA(y,
                    nas)), na.method, method == "kendall")
            }
        }
        else if (na.method != 3L) {
            x <- Rank(x)
            if (!is.null(y))
                y <- Rank(y)
            .Call(C_cor, x, y, na.method, method == "kendall")
        }
        else {
            if (is.null(y)) {
                ncy <- ncx <- ncol(x)
                if (ncx == 0)
                    stop("'x' is empty")
                r <- matrix(0, nrow = ncx, ncol = ncy)
                for (i in seq_len(ncx)) {
                    for (j in seq_len(i)) {
                      x2 <- x[, i]
                      y2 <- x[, j]
                      ok <- complete.cases(x2, y2)
```

```
                  x2 <- rank(x2[ok])
                  y2 <- rank(y2[ok])
                  r[i, j] <- if (any(ok))
                    .Call(C_cor, x2, y2, 1L, method == "kendall")
                  else NA
                }
            }
            r <- r + t(r) - diag(diag(r))
            rownames(r) <- colnames(x)
            colnames(r) <- colnames(x)
            r
        }
        else {
            if (length(x) == 0L || length(y) == 0L)
                stop("both 'x' and 'y' must be non-empty")
            matrix_result <- is.matrix(x) || is.matrix(y)
            if (!is.matrix(x))
                x <- matrix(x, ncol = 1L)
            if (!is.matrix(y))
                y <- matrix(y, ncol = 1L)
            ncx <- ncol(x)
            ncy <- ncol(y)
            r <- matrix(0, nrow = ncx, ncol = ncy)
            for (i in seq_len(ncx)) {
                for (j in seq_len(ncy)) {
                  x2 <- x[, i]
                  y2 <- y[, j]
                  ok <- complete.cases(x2, y2)
                  x2 <- rank(x2[ok])
                  y2 <- rank(y2[ok])
                  r[i, j] <- if (any(ok))
                    .Call(C_cor, x2, y2, 1L, method == "kendall")
                  else NA
                }
            }
            rownames(r) <- colnames(x)
            colnames(r) <- colnames(y)
            if (matrix_result)
                r
            else drop(r)
        }
    }
}
Browse[2]>
debug: na.method <- pmatch(use, c("all.obs", "complete.obs", "pairwise.complete.obs",
    "everything", "na.or.complete"))
Browse[2]>
debug: if (is.na(na.method)) stop("invalid 'use' argument")
Browse[2]>
debug: method <- match.arg(method)
Browse[2]>
debug: if (is.data.frame(y)) y <- as.matrix(y)
Browse[2]>
debug: if (is.data.frame(x)) x <- as.matrix(x)
Browse[2]>
debug: if (!is.matrix(x) && is.null(y)) stop("supply both 'x' and 'y' or a matrix-like 'x'")
Browse[2]>
```

```
debug: if (!(is.numeric(x) || is.logical(x))) stop("'x' must be numeric")
Browse[2]>
debug: stopifnot(is.atomic(x))
Browse[2]>
debug: if (!is.null(y)) {
    if (!(is.numeric(y) || is.logical(y)))
        stop("'y' must be numeric")
    stopifnot(is.atomic(y))
}
Browse[2]>
debug: if (!(is.numeric(y) || is.logical(y))) stop("'y' must be numeric")
Browse[2]>
debug: stopifnot(is.atomic(y))
Browse[2]>
debug: Rank <- function(u) {
    if (length(u) == 0L)
        u
    else if (is.matrix(u)) {
        if (nrow(u) > 1L)
            apply(u, 2L, rank, na.last = "keep")
        else row(u)
    }
    else rank(u, na.last = "keep")
}
Browse[2]>
debug: if (method == "pearson") .Call(C_cor, x, y, na.method, FALSE) else if (na.method %in%
    c(2L, 5L)) {
    if (is.null(y)) {
        .Call(C_cor, Rank(na.omit(x)), NULL, na.method, method ==
            "kendall")
    }
    else {
        nas <- attr(na.omit(cbind(x, y)), "na.action")
        dropNA <- function(x, nas) {
            if (length(nas)) {
                if (is.matrix(x))
                  x[-nas, , drop = FALSE]
                else x[-nas]
            }
            else x
        }
        .Call(C_cor, Rank(dropNA(x, nas)), Rank(dropNA(y, nas)),
            na.method, method == "kendall")
    }
} else if (na.method != 3L) {
    x <- Rank(x)
    if (!is.null(y))
        y <- Rank(y)
    .Call(C_cor, x, y, na.method, method == "kendall")
} else {
    if (is.null(y)) {
        ncy <- ncx <- ncol(x)
        if (ncx == 0)
            stop("'x' is empty")
        r <- matrix(0, nrow = ncx, ncol = ncy)
        for (i in seq_len(ncx)) {
            for (j in seq_len(i)) {
                x2 <- x[, i]
                y2 <- x[, j]
                ok <- complete.cases(x2, y2)
                x2 <- rank(x2[ok])
                y2 <- rank(y2[ok])
                r[i, j] <- if (any(ok))
                  .Call(C_cor, x2, y2, 1L, method == "kendall")
                else NA
            }
        }
        r <- r + t(r) - diag(diag(r))
        rownames(r) <- colnames(x)
        colnames(r) <- colnames(x)
        r
    }
    else {
        if (length(x) == 0L || length(y) == 0L)
            stop("both 'x' and 'y' must be non-empty")
        matrix_result <- is.matrix(x) || is.matrix(y)
        if (!is.matrix(x))
            x <- matrix(x, ncol = 1L)
        if (!is.matrix(y))
            y <- matrix(y, ncol = 1L)
        ncx <- ncol(x)
        ncy <- ncol(y)
        r <- matrix(0, nrow = ncx, ncol = ncy)
        for (i in seq_len(ncx)) {
            for (j in seq_len(ncy)) {
                x2 <- x[, i]
                y2 <- y[, j]
                ok <- complete.cases(x2, y2)
                x2 <- rank(x2[ok])
                y2 <- rank(y2[ok])
                r[i, j] <- if (any(ok))
                  .Call(C_cor, x2, y2, 1L, method == "kendall")
                else NA
            }
        }
        rownames(r) <- colnames(x)
        colnames(r) <- colnames(y)
        if (matrix_result)
            r
        else drop(r)
    }
}
Browse[2]>
debug: .Call(C_cor, x, y, na.method, FALSE)
Browse[2]> # Press Ctrl + c
Program received signal SIGINT, Interrupt.
0x00007ffff62cbd83 in __select_nocancel ()
    at ../sysdeps/unix/syscall-template.S:81
81     ../sysdeps/unix/syscall-template.S: No such file or directory.
(gdb)
(gdb) b corcov
Breakpoint 1 at 0x7ffff46167c0: file cov.c, line 637.
(gdb) c
Continuing.
```

```
# Press Return key

Breakpoint 1, corcov (x=0x110b348, y=0x153bf00, na_method=0xb17d18,
    skendall=0xc88788, cor=TRUE) at cov.c:637
637    {
(gdb) backtrace
#0  corcov (x=<optimized out>, y=0x153bf00, na_method=<optimized out>,
    skendall=<optimized out>, cor=TRUE) at cov.c:741
#1  0x0000000000480120 in do_dotcall (call=0xe36030, op=<optimized out>,
    args=<optimized out>, env=<optimized out>) at dotcode.c:1251
#2  0x00000000004bd03f in Rf_eval (e=0xe36030, rho=0xcefcf0) at eval.c:655
#3  0x00000000004bcea9 in Rf_eval (e=0xe36c18, rho=rho@entry=0xcefcf0)
    at eval.c:627
#4  0x00000000004befd9 in do_begin (call=0xe3eb58, op=0x97c678,
    args=0xe36be0, rho=0xcefcf0) at eval.c:1716
#5  0x00000000004bcea9 in Rf_eval (e=0xe3eb58, rho=0xcefcf0) at eval.c:627
#6  0x00000000004be0ff in Rf_applyClosure (call=call@entry=0xcf2df0,
    op=op@entry=0xe3f668, arglist=0xcefa50, rho=rho@entry=0x9a5e68,
    suppliedvars=0x96f928) at eval.c:1039
#7  0x00000000004bccd9 in Rf_eval (e=e@entry=0xcf2df0, rho=rho@entry=0x9a5e68)
    at eval.c:674
#8  0x00000000004e2041 in Rf_ReplIteration (rho=rho@entry=0x9a5e68,
    savestack=savestack@entry=0, browselevel=browselevel@entry=0,
    state=state@entry=0x7fffffffcdc0) at main.c:258
#9  0x00000000004e2368 in R_ReplConsole (rho=0x9a5e68, savestack=0,
    browselevel=0) at main.c:308
#10 0x00000000004e2411 in run_Rmainloop () at main.c:1006
#11 0x00000000004e2452 in Rf_mainloop () at main.c:1013
#12 0x0000000000418b78 in main (ac=ac@entry=2, av=av@entry=0x7fffffffdef8)
---Type <return> to continue, or q <return> to quit---
    at Rmain.c:29
#13 0x00007ffff61fbec5 in __libc_start_main (main=0x418b60 <main>, argc=2,
    argv=0x7fffffffdef8, init=<optimized out>, fini=<optimized out>,
    rtld_fini=<optimized out>, stack_end=0x7fffffffdee8) at libc-start.c:287
#14 0x0000000000418ba8 in _start ()

(gdb) next
643        if(isNull(x)) /* never allowed */
.....
771        if (ansmat) { /* set dimnames() when applicable */
(gdb)
797        if(sd_0)/* only in cor() */
(gdb)
799        UNPROTECT(nprotect);
(gdb)
801    }
(gdb)
do_dotcall (call=0xe36030, op=<optimized out>, args=<optimized out>,
    env=<optimized out>) at dotcode.c:1252
1252       vmaxset(vmax);
(gdb)
1251       retval = R_doDotCall(ofun, nargs, cargs, call);
(gdb)
1252       vmaxset(vmax);
(gdb)
1254   }
(gdb)
```

```
Rf_eval (e=0xe36030, rho=0xcefcf0) at eval.c:656
656                 R_Srcref = oldref;
(gdb)
657                 endcontext(&cntxt);
(gdb)
656                 R_Srcref = oldref;
...
(gdb)
Rf_eval (e=0xe36c18, rho=rho@entry=0xcefcf0) at eval.c:637
637              if (flag < 2) R_Visible = flag != 1;
(gdb)
669              check_stack_balance(op, save);
...
do_begin (call=0xe3eb58, op=0x97c678, args=0xe36be0, rho=0xcefcf0)
    at eval.c:1709
1709         while (args != R_NilValue) {
(gdb)
1718             args = CDR(args);
....
Rf_eval (e=0xe3eb58, rho=0xcefcf0) at eval.c:637
637              if (flag < 2) R_Visible = flag != 1;
(gdb)
669              check_stack_balance(op, save);
...
(gdb)
Rf_applyClosure (call=call@entry=0xcf2df0, op=op@entry=0xe3f668,
    arglist=0xcefa50, rho=rho@entry=0x9a5e68, suppliedvars=0x96f928)
    at eval.c:1042
1042      endcontext(&cntxt);
...
1045         Rprintf("exiting from: ");
(gdb)
exiting from: 1046        PrintValueRec(call, rho);
(gdb)
cor(1:5, rnorm(5))
1048      UNPROTECT(3);
...
Rf_eval (e=e@entry=0xcf2df0, rho=rho@entry=0x9a5e68) at eval.c:675
675              UNPROTECT(1);
(gdb)
...
Rf_eval (e=e@entry=0xcf2df0, rho=rho@entry=0x9a5e68) at eval.c:675
675              UNPROTECT(1);
(gdb)
...
Rf_ReplIteration (rho=rho@entry=0x9a5e68, savestack=savestack@entry=0,
    browselevel=browselevel@entry=0, state=state@entry=0x7fffffffcdc0)
    at main.c:259
259          SET_SYMVALUE(R_LastvalueSymbol, value);
260          wasDisplayed = R_Visible;
(gdb)
261          if (R_Visible)
(gdb)
262              PrintValueEnv(value, rho);
(gdb)
[1] -0.6156519
...
```

```
R_ReplConsole (rho=0x9a5e68, savestack=0, browselevel=0) at main.c:309
309	        if(status < 0) {
(gdb)
308	        status = Rf_ReplIteration(rho, savestack, browselevel, &state);
(gdb)
>
309	        if(status < 0) {
(gdb)
308	        status = Rf_ReplIteration(rho, savestack, browselevel, &state);
(gdb)
>
[Inf loop]
> q('no')
[Inferior 1 (process 25390) exited normally]
(gdb) quit
brb@brb-T3500:~/Downloads/R-3.2.2$
```