2. Debugging Tools

Contents

1 2. Debugging Tools

1.1 Debugging Tools - Diagnosing the Problem

1.1.1 Something's Wrong!

Indications that something's not right · message: A generic notification/diagnostic message produced by the message function; execution of the function continues · warning: An indication that something is wrong but not necessarily fatal; execution of the function continues; generated by the warning function · error: An indication that a fatal problem has occurred; execution stops; produced by the stop function · condition: A generic concept for indicating that something unexpected can occur; programmers can create their own conditions

Warning

```
log(-1)
```

```
## Warning in log(-1): NaNs produced
```

```
## [1] NaN
```

```
printmessage <- function(x) {
  if (x > 0)
    print("x is greater than zero")
  else
    print("x is less than or equal to zero")
  invisible(x)
}

printmessage(1)
```

```
## [1] "x is greater than zero"
```

The invisible() function in R is used to prevent a value from being automatically printed when a function is called interactively, while still allowing the value to be returned and used in subsequent operations.

```
printmessage(NA)
```

```
## Error in if (x > 0) print("x is greater than zero") else print("x is less than or equal to zero"): mi
```

```
printmessage2 <- function(x) {
  if (is.na(x))
    print("x is a missing value!")
  else if (x > 0)
    print("x is greater than zero")
  else
    print("x is less than or equal to zero")
  invisible(x)
}
```

```
x <- log(-1)
```

```
## Warning in log(-1): NaNs produced
```

```
printmessage2(x)
```

```
## [1] "x is a missing value!"
```

How do you know that something is wrong with your function? · What was your input? How did you call the function? · What were you expecting? Output, messages, other results? · What did you get? · How does what you get differ from what you were expecting? · Were your expectations correct in the first place? · Can you reproduce the problem (exactly)?

1.2 Debugging Tools in R

The primary tools for debugging functions in R are · *traceback*: prints out the function call stack after an error occurs; does nothing if there's no error · *debug*: flags a function for "debug" mode which allows you to step through execution of a function one line at a time · *browser*: suspends the execution of a function wherever it is called and puts the function in debug mode · *trace*: allows you to insert debugging code into a function a specific places · *recover*: allows you to modify the error behavior so that you can browse the function call stack These are interactive tools specifically designed to allow you to pick through a function. There's also the more blunt technique of inserting print/cat statements in the function.

1.3 Debugging Tools - Using the Tools

1.3.1 traceback

```
mean(x)
```

```
## [1] NaN
```

```
traceback()
```

```
## No traceback available
```

```
lm(y~x)
```

```
## Error in eval(predvars, data, env): object 'y' not found
```

```
traceback()
```

```
## No traceback available
```

1.4 debug

```
debug(lm)
```

```
lm(y~x)
```

```
## debugging in: lm(y ~ x)
## debug: {
##     ret.x <- x
##     ret.y <- y
##     cl <- match.call()
##     mf <- match.call(expand.dots = FALSE)
##     m <- match(c("formula", "data", "subset", "weights", "na.action",
##         "offset"), names(mf), 0L)
##     mf <- mf[c(1L, m)]
##     mf$drop.unused.levels <- TRUE
##     mf[[1L]] <- quote(stats::model.frame)
##     mf <- eval(mf, parent.frame())
##     if (method == "model.frame")
##         return(mf)
##     else if (method != "qr")
##         warning(gettextf("method = '%s' is not supported. Using 'qr'",
##             method), domain = NA)
##     mt <- attr(mf, "terms")
##     y <- model.response(mf, "numeric")
##     w <- as.vector(model.weights(mf))
##     if (!is.null(w) && !is.numeric(w))
##         stop("'weights' must be a numeric vector")
##     offset <- model.offset(mf)
##     mlm <- is.matrix(y)
##     ny <- if (mlm)
##         nrow(y)
##     else length(y)
##     if (!is.null(offset)) {
##         if (!mlm)
##             offset <- as.vector(offset)
##         if (NROW(offset) != ny)
##             stop(gettextf("number of offsets is %d, should equal %d (number of observations)",
##                 NROW(offset), ny), domain = NA)
##     }
##     if (is.empty.model(mt)) {
##         x <- NULL
##         z <- list(coefficients = if (mlm) matrix(NA_real_, 0,
##             ncol(y)) else numeric(), residuals = y, fitted.values = 0 *
##             y, weights = w, rank = 0L, df.residual = if (!is.null(w)) sum(w !=
##             0) else ny)
##         if (!is.null(offset)) {
##             z$fitted.values <- offset
##             z$residuals <- y - offset
##         }
##     }
##     else {
##         x <- model.matrix(mt, mf, contrasts)
##         z <- if (is.null(w))
##             lm.fit(x, y, offset = offset, singular.ok = singular.ok,
##                 ...)
##         else lm.wfit(x, y, w, offset = offset, singular.ok = singular.ok,
##             ...)
```

```
##     }
##     class(z) <- c(if (mlm) "mlm", "lm")
##     z$na.action <- attr(mf, "na.action")
##     z$offset <- offset
##     z$contrasts <- attr(x, "contrasts")
##     z$xlevels <- .getXlevels(mt, mf)
##     z$call <- cl
##     z$terms <- mt
##     if (model)
##         z$model <- mf
##     if (ret.x)
##         z$x <- x
##     if (ret.y)
##         z$y <- y
##     if (!qr)
##         z$qr <- NULL
##     z
## }
## debug: ret.x <- x
## debug: ret.y <- y
## debug: cl <- match.call()
## debug: mf <- match.call(expand.dots = FALSE)
## debug: m <- match(c("formula", "data", "subset", "weights", "na.action",
##     "offset"), names(mf), 0L)
## debug: mf <- mf[c(1L, m)]
## debug: mf$drop.unused.levels <- TRUE
## debug: mf[[1L]] <- quote(stats::model.frame)
## debug: mf <- eval(mf, parent.frame())
```

```
## Error in eval(predvars, data, env): object 'y' not found
```

1.4.1 recover

```
options(error = error)
```

```
## Error in eval(expr, envir, enclos): object 'error' not found
```

```
read.csv("nosuchfile")
```

```
## Warning in file(file, "rt"): cannot open file 'nosuchfile': No such file or
## directory
```

```
## Error in file(file, "rt"): cannot open the connection
```

1.4.2 Debugging

Summary · There are three main indications of a problem/condition: message, warning, error - onlyanerrorisfatal · When analyzing a function with a problem, make sure you can reproduce the problem, clearly state your expectations and how the output differs from your expectation · Interactive debugging tools traceback, debug, browser, trace, and recover can be used to find problematic code in functions · Debugging tools are not a substitute for thinking!