

# NEWS for pqR version 2.15.0 (2016-06-24)

---

NEWS

*pqR News*

---

## CHANGES IN VERSION RELEASED 2016-06-24

### INTRODUCTION:

- This release extends the R language in ways that address a set of related flaws in the design of R, and before it S.  
These extensions make it easier to write reliable programs, by making the easy way to do things also be the correct way, unlike the previous situation with sequence generation using the colon operator, and dimension dropping when subsetting arrays.
- Several other changes in features are also implemented in this version, some of which are related to the major language extensions.
- There are also a few bug fixes, and some improvements in testing, but no major performance improvements (though some tweaks).

### MAJOR LANGUAGE EXTENSIONS AND OTHER CHANGES:

- There is a new `..` operator for generating increasing integer sequences, which is a less error-prone replacement for the `:` operator (which remains for backwards compatibility). Since `..` generates only increasing sequences, it can generate an empty sequence when the end value is less than the start value, thereby avoiding some very common bugs that arise when `:` is used.

The `..` operator also has lower precedence than arithmetic operators (unlike `:`), which avoids another common set of bugs.

For example, the following code sets all interior elements of the matrix `M` to zero, that is, all elements except those in the first or last row or column:

```
for (i in 2..nrow(M)-1)
  for (j in 2..ncol(M)-1)
    M[i,j] <- 0
```

Without the new `..` operator, it is awkward to write code for this task that works correctly when `M` has two or fewer rows, or two or fewer columns.

- In order that the `..` operator can be conveniently used in contexts such as `i..j`, consecutive dots are no longer allowed in names (without using backticks), except at the beginning or end. So `i..j` is not a valid name, but `..i..` is valid (though not

recommended). With this restriction on names, most uses of the `..` operator are unambiguous even if it is not surrounded by spaces. The only exceptions are some uses in which `..` is written with a space after it but not before it, expressions such as `i..(a+b)`, which is a call of a function named `i..`, and expressions such as `i..-j`, which returns the difference between `i..` and `j`. Most such uses will be stylistically bad, redundant (note that the parentheses around `a+b` above are unnecessary), or probably unlikely (as is the case for `i..-j`).

To accomodate old R code that has consecutive dots within names, parsing of the `..` operator can be disabled by setting the `parse_dotdot` option to `FALSE` (with the `options` function). The `parse_dotdot` option defaults to `TRUE` unless the environment variable `R_PARSE_DOTDOT` is set to `FALSE`. When `parse_dotdot` is `FALSE`, consecutive dots are allowed in names, and `..` is not a reserved word.

- Another source of bugs is the automatic dropping of dimensions of size one when subsetting matrices (or higher-dimensional arrays) using `[]`, unless the `drop=FALSE` argument is specified. This frequently results in code that mostly works, but not when, for example, a data set has only one observation, or a model uses only one explanatory variable.

To make handling this problem easier, if no `drop` argument is specified, `pqR` now does not drop a dimension of size one if the subscript for that dimension is a one-dimensional non-logical array. For example, if `A` is a matrix, `A[1..100,array(1)]` will produce a matrix, whereas `A[1..100,1]` will produce a vector.

To make this feature more useful, the new `..` operator produces a one-dimensional array, not a bare vector. So `A[1..n,1..m]` will always produce a matrix result, even when `n` or `m` are one. (It will also correctly produce an array with zero rows or zero columns when `n` or `m` are zero.)

This change also applies to subsetting of data frames. For example, `df[1..10,1..n]` will return a data frame (not a vector) even when `n` is one.

- Problems with dimensions of size one being dropped also arise when an entire row, or an entire column, is selected with an empty (missing) subscript, and there happens to be only one row, or only one column. For example, if `A` is a matrix with one column, `A[1:10,]` will be a vector, not a matrix.

To address this problem, `pqR` now allows a missing argument to be specified by `_`, rather than by nothing at all, and the `[]` operator (for matrices, arrays, and data frames) will not drop a dimension if its subscript is `_`. So `A[1:10,_]` will be a matrix even when `A` has only one column.

R functions that check for a missing argument with the `missing` function will see both an empty argument and `_` as missing, but can distinguish them using the `missing_from_underline` function.

- A common use of `for` statements is to iterate over indexes of a vector, or row and column indexes of a matrix. A new type of `for` statement with “along” rather than “in” now makes this more convenient.

For vectors, the form

```
for (i along vec) ...
```

is equivalent to

```
for (i in seq_along(vec)) ...
```

For matrices, the form

```
for (i, j along M) ...
```

is equivalent to

```
for (j in 1..ncol(M))
  for (i in 1..nrow(M))
    ...
```

However, if `M` is of a class with its own `dim` method, this method is not used (effectively, `ncol(unclass(M))` and `nrow(unclass(m))` are used). This may well change in future, and similarly a `length` method may in future be used when “along” is used with a vector.

- Because of the new restriction on names, the `make.names` function will now (by default) convert a sequence of consecutive dots in the name it would otherwise have made to a single dot. (See `help(make.names)` for further details).
- For the same reason, `make.unique` has been changed so that the separator string (which defaults to a dot) will not be appended to a name if the name already ends in that string.

#### BUG FIXES:

- Fixed a bug (or mis-feature) in subsetting with a single empty subscript, as in `A[]`. This now works the same as if the empty subscript had been the sequence of all indexes (ie, like `A[1..length(A)]`), which removes all attributes except names. R Core versions to at least R-3.3.1 instead return `A` unchanged, preserving all attributes, though attributes are not retained with other uses of the `[]` operator. This is contrary to the description in `help("[")`, and also does not coincide with the (different) description in the R language definition. Returning `A` unchanged is not only inconsistent, but also useless, since there is then no reason to ever write `A[]`. However, internally, R Core implementations duplicate `A`, which may be of significance when `A[]` is passed as an argument of `.C`, `.Fortran`, `.Call`, or `.External`, but only if the programmer is not abiding by the rules. However, in `pqR`, the data part of a vector or matrix is still copied when `A[]` is evaluated, so such rule-breaking should still largely be accommodated.
- Fixed bugs in the conversion of strings to numbers, so that the behaviour now matches `help(NumericConstants)`, which states that numeric constants are parsed very similarly to C99. This was not true before (or in R-2.15.0) – some erroneous syntax was accepted without error, and some correct syntax was rejected, or gave the wrong value. In particular, fractional parts are now accepted for hexadecimal constants. Later R Core versions made some fixes, but up to at least R-3.3.1 there are still problems. For example, in R-3.3.1, `parse(text="0x1.8")[[1]]` gives an error, and `as.numeric("0x1.8")` produces 24 (as does `scan` when given this input). In this version of `pqR`, these return the correct value of 1.5.
- Fixed a problem with identifying the version of the `makeinfo` program that is installed that arises with recent versions of `makeinfo`.
- Put in a check for non-existent primitives when unserializing R objects, as was done in R-3.0.1.
- Fixed a bug (also in R-2.15.0, but fixed in later R Core versions) illustrated by the following code:

```
a <- array(c(3,4),dimnames=list(xyz=c("fred","bert")))
print(a[1:2])
print(a[]) # should print same thing, but didn't
```

- Fixed a bug illustrated by the following code:

```
f <- function (x) { try(x); missing(x) }
g <- function (y) f(y)
h <- function (z) g(z)

f(pi[1,1]) # FALSE
g(pi[1,1]) # FALSE
h(pi[1,1]) # Should also be FALSE, but isn't!
```

This bug is in R Core versions to at least R-3.3.1.

- Fixed a bug in which an internal error message is displayed as shown below:

```
> f <- function (...) ..1; f()
Error in f() : 'nthcdr' needs a list to CDR down
```

A sensible error message is now produced. This bug is also in R Core versions to at least R-3.3.1.

- Fixed a bug in S4 method dispatch that caused failure of the no-segfault test done by make check-all on Windows 10 (pqR issue #29 + related fix). (Also in R-2.15.0, and partially fixed in R-3.3.0.)
- Fixed a bug illustrated by
 

```
atan; show <- function (x) cat("HI\n"); atan
```

 Now, pqR no longer prints HI! for the second display of `atan`.
- Fixed a pqR bug in which the result of `getParseData` omitted the letter at the end of `1i` or `1L`.
- Fixed a pqR bug in which enabling trace output from the helpers module and then typing control/C while trace output is being printed could lead to pqR hanging.

## CHANGES IN VERSION RELEASED 2015-09-14

### INTRODUCTION:

- With this release, pqR now works on Microsoft Windows systems. See below for details.
- The facilities for embedding R in other applications have also been tested in this release, and some problems with how this is done in R Core versions have been fixed.
- The parser and deparser, and the method for performing the basic Read-Eval-Print Loop, have been substantially rewritten. This has few user-visible effects at present (apart from bug fixes and performance improvements), but sets the stage for future improvements in pqR.
- The facility for recording detailed parsing data introduced in R-3.0.0 has now been implemented in pqR as part of the parser rewrite.
- There are also a few other improvements and bug fixes.

### INSTALLATION ON MICROSOFT WINDOWS:

- Building pqR on Microsoft Windows systems, using the Rtools facilities, has now been tested, and some problems found in this environment have been fixed. Binary distributions are not yet provided, however.
- Detailed and explicit instructions for building pqR from source on Windows systems are now provided, in the `'src/gnuwin32/INSTALL'` file of the pqR source directory.

These instructions mostly correspond to information in The R Installation and Administration manual, but in more accessible form.

- See [pqR-project.org](http://pqR-project.org) for more information on Windows systems on which pqR has been tested, and on any problems and workarounds that may have been discovered.
- The Writing R Extensions manual now warns that on Windows, with the Rtools toolchain, a thread started by OpenMP may have its floating point unit set so that long double arithmetic is the same as double arithmetic. Use `__asm__("fninit")` in C to reset the FPU so that long double arithmetic will work.
- The default is now to install packages from source, since there is no binary repository for pqR.

#### EMBEDDED R FACILITIES AND EXAMPLES:

- The `R_ReplDLLinit` and `R_ReplDLLdo1` functions in `'src/main/main.c'` have been fixed to handle errors correctly, and to avoid code duplication with `R_ReplIteration`.
- Another test of embedded R has been added to `'tests/Embedding'`, which is the same as an example in the R Extensions manual, which has been improved.
- Another example in the R Extensions manual has been changed to mimic `'src/gnuwin32/embeddedR.c'`.
- The example in `'src/gnuwin32/front-ends/rtest.c'` has also been updated.

#### DOCUMENTATION UPDATES:

- The R Language Definition and the help files on assignment operators (eg, `help("=")`) contained incorrect and incomplete information on the precedence of operators, especially the assignment operators. This and other incorrect information has been corrected.
- The examples in `help(parse)` and `help(getParseData)` have been improved.

#### INTERNAL CODE REWRITES:

- The parser has been rewritten to use top-down recursive descent, rather than a bottom-up parser produced by Bison as was used previously. This substantially simplifies the parser, and allows several kludges in the previous scheme to be eliminated. Also, the rewritten parser can now record detailed parse information (see below). The new parser for pqR is usually about a factor of 1.5 faster than the parser in R-3.2.2, but it is sometimes enormously faster, since the parser in R-3.2.2 will in some contexts take time growing as the square of the length of the source file.
- Much of the deparser has been rewritten. It no longer looks at the definitions of operators, which are irrelevant, since the parser does not look at them.
- The methods by which the Read-Eval-Print Loop (REPL) is done (in various contexts) have been rationalized, in coordination with the new parsing scheme.

#### NEW FEATURES:

- In pqR-2015-07-11, the parser was changed to not include parentheses in R language objects if they were necessary in order for the expression to be parsed correctly. Omitting such parentheses improves performance. In this version, such parentheses are removed only if the `keep.parens` option is `FALSE` (the default). Also, parentheses are never removed from expressions that are on the right side of a formula, since some packages assign significance to such parentheses beyond their grouping function.

- The right assignment operators, `->` and `->>`, are now real operators. Previously (and in current R Core versions), expressions involving these operators were converted to the corresponding left assignment expressions. This has the potential to cause pointless confusion.
- The `**` operator, which has always been accepted as a synonym for the `^` operator, is now recorded as itself, rather than being converted to `^` by the parser. This avoids unnecessary anomalies such as the following confusing error report:

```
> a - **b
Error: unexpected '^' in "a - **"
```

The `**` operator is defined to be the same primitive as `^`, which is associated with the name `^`, and hence dispatches on methods for `^` even if called via `**`.

#### NEW FEATURES FROM LATER R CORE VERSIONS:

- From R-3.0.0: For compatibility with packages written to be able to handle the long vectors introduced in R-3.0.0, definitions for `R_xlen_t`, `R_XLEN_T_MAX`, `XLENGTH`, `XTRUELENGTH`, `SHORT_VEC_LENGTH`, `SET_SHORT_VEC_TRUELENGTH` are now provided, all the same as the corresponding regular versions (as is also the case for R-3.0.0+ on 32-bit platforms). The `IS_LONG_VEC` macro is also defined (as always false). Note, however, that packages that declare a dependency on `R >= 3.0.0` will not install even if they would in fact work with pqR because of these compatibility definitions.
- From R-3.0.0: The `srcfile` argument to `parse()` may now be a character string, to be used in error messages.
- The facilities for recording detailed parsing information from R-3.0.0 are now implemented in pqR, as part of the rewrite of the parser, along with the extension to provide partial parse information when a syntax error occurs that was introduced in R-3.0.2. See help on `parse` and `getParseData` for details.
- From R-2.15.2: On Windows, the C stack size has been increased to 64MB (it has been 10MB since the days of 32MB RAM systems).

#### PERFORMANCE IMPROVEMENTS:

- Character-at-a time input has been sped up by reducing procedure call overhead. This significantly speeds up `readLines` and `scan`.
- The new parser is faster than the old parser, both because of the parser rewrite (see above) and because of the faster character input.

#### BUG FIXES MATCHING THOSE IN LATER R CORE VERSIONS:

- From R-2.15.1: Names containing characters which need to be escaped were not deparsed properly (PR#14846). Fixed in pqR partly based on R Core fix.
- From R-2.15.2: When given a 0-byte file and asked to keep source references, `parse()` read input from `stdin()` instead.
- From R-2.15.3: Expressions involving user defined operators were not always deparsed faithfully (PR#15179). Fixed in pqR as part of the rewrite of the parser and deparser.
- From R-3.0.2: `source()` did not display filenames when reporting syntax errors.
- From R-3.1.3: The parser now gives an error if a null character is included in a string using Unicode escapes. (PR#16046)

- From R-3.0.2: Deparsing of infix operators with named arguments is improved (PR#15350). [ In fact, the change, both in pqR and in R Core versions, is only with respect to operators in percent signs, such as `%fred%`, with these now being deparsed as function calls if either argument is named. ]
- From R-3.2.2: Rscript and command line R silently ignored incomplete statements at the end of a script; now they are reported as parse errors (PR#16350). Fixed in pqR as part of the rewrite of the parser and deparser.
- From R-3.2.1: The parser could overflow internally when given numbers in scientific format with extremely large exponents. (PR#16358). Fixed in pqR partly as in R Core fix. Was actually a problem with any numerical input, not just with the parser.
- From R-3.1.3: Extremely large exponents on zero expressed in scientific notation (e.g. `0.0e50000`) could give NaN (PR#15976). Fixed as in R Core fix.
- From R-2.15.3: On Windows, work around an event-timing problem when the RGui console was closed from the 'X' control and the closure cancelled. (This would on some 64-bit systems crash R, typically those with a slow GPU relative to the CPU.)

#### BUG FIXES:

- Fixed a bug in which a "cons memory exhausted" error could be raised even though a full garbage collection that might recover more memory had not been attempted. (This bug appears to be present in R Core versions as well.)
- The new parser fixes bugs arising from the old parser's kludge to handle semicolons, illustrated by the incorrect output seen below:

```
> p<-parse()
?"abc;xyz"
Error in parse() : <stdin>:1:1: unexpected INCOMPLETE_STRING
1: "abc;
  ^

> p<-parse()
?8 #abc;xyz
Error in parse() : <stdin>:1:7: unexpected end of input
1: 8 #abc;
  ^
```

- Fixed deparsing of complex numbers, which were always deparsed as the sum of a real and an imaginary part, even though the parser can only produce complex numbers that are pure imaginary. For example, the following output was produced before:

```
> deparse(quote(3*5i))
[1] "3 * (0+5i)"
```

This is now deparsed to `"3 * 5i"`. This bug exists in all R Core versions through at least R-3.2.2.

- Fixed a number of bugs in the deparser that are illustrated by the following, which produce incorrect output as noted, in R Core versions through at least R-3.2.2:

```
deparse(parse(text="`+`(a,b)[1]")[1]) # Omits necessary parens
deparse(quote(`[<-`(x,1)),control="S_compatible") # unmatched " and '
deparse(parse(text="a = b <- c")[1]) # Puts in unnecessary parens
deparse(parse(text="a+!b")[1]) # Puts in unnecessary parens
deparse(parse(text="?lm")[1]) # Doesn't know about ? operator
deparse(parse(text="a:=b")[1]) # Doesn't know about := operator
```

```

deparse(parse(text="a$x'")[[1]])      # Conflates name and character
deparse(parse(text="`*(2)")[[1]])    # Result is syntactically invalid
deparse(parse(text="`$(a,b+2)")[[1]]) # Result is syntactically invalid
e<-quote(if(x) X else Y); e[[3]]<-quote(if(T)3); deparse(e) # all here
e <- quote(f(x)); e[[2]] <- quote((a=1))[[2]]; deparse(e) # and below
e <- quote(f(Q=x)); e[[2]] <- quote((a=1))[[2]]; deparse(e) # need parens
e <- quote(while(x) 1); e[[2]] <- quote((a=1))[[2]]; deparse(e)
e <- quote(if(x) 1 else 2); e[[2]] <- quote((a=1))[[2]]; deparse(e)
e <- quote(for(x in y) 1); e[[3]] <- quote((a=1))[[2]]; deparse(e)

```

In addition, the bug illustrated below was fixed, which was fixed (differently) in R-3.0.0:

```
a<-quote(f(1,2)); a[[1]]<-function(x,y)x+y; deparse(a) # Omits parens
```

- Fixed the following bug (also in R Core versions to at least R-3.2.2):

```

> parse()
?'\12a\x.'
Error: '\x' used without hex digits in character string starting "'\1a\x"

```

Note that the "2" has disappeared from the error message. This bug also affected the results of `getParseData`.

- Fixed a memory leak that can be seen by running the code below:

```

> long <- paste0 (c(' ', rep("1234567890",820), '\x.'), collapse="")
> for (i in 1:1000000) try (e <- parse(text=long), silent=TRUE)

```

The leak will not occur if 820 is changed to 810 in the above. This bug also exists in R Core versions to at least R-3.2.2.

- Entering a string constant containing Unicode escapes that was 9999 or 10000 characters long would produce an error message saying "String is too long (max 10000 chars)". This has been fixed so that the maximum now really is 10000 characters. (Also present in R Core versions, to at least R-3.2.2.)
- Fixed a bug that caused the error caret in syntax error reports to be misplaced when more than one line of context was shown. This was supposedly fixed in R-3.0.2, but incorrectly, resulting in the error caret being misplaced when only one line of context is shown (in R Core versions to at least R-3.2.2).
- On Windows, running R.exe from a command prompt window would result in Ctrl-C misbehaving. This was [PR#14948](#) at R Core, which was supposedly fixed in R-2.15.2, but the fix only works if a 32 or 64 bit version of R.exe is selected manually, not if the version of R.exe that automatically runs the R.exe for a selected architecture is used (which is the intended normal usage).

## CHANGES IN VERSION RELEASED 2015-07-11

### INTRODUCTION:

- This version is a minor modification of the version of pqR released on 2015-06-24, which does not have a separate NEWS section, incorporating also the changes in the version released 2015-07-08. These modifications fix some installation and testing issues that caused problems on some platforms. There are also a few documentation and bug fixes, a few more tests, and some expansion in the use of static boxes (see below). Version 2015-06-24 of pqR improved reliability and portability, and also contained some performance improvements, including some that substantially speed



up interpretive execution of programs that do many scalar operations. Details are below.

#### INSTALLATION:

- The method used to quickly test for NaN/NA has changed to one that should work universally for all current processors (any using IEEE floating point, as already assumed in R, with consistent endianness, as is apparently the case for all current general-purpose processors, and was partially assumed before). There is therefore no longer any reason to define the symbol `ENABLE_ISNAN_TRICK` when compiling pqR (it will be ignored if defined).
- The module used to support parallel computation in helper threads has been updated to avoid a syntactic construction that technically violates the OpenMP 3.1 specification. This construction had been accepted without error by gcc 4.8 and earlier, but is not accepted by some recent compilers.
- The tests in the version supplied of the recommended Matrix package have been changed to not assume things that may not be true regarding the speed and long double precision of the machine being used. (These tests produced spurious errors on some platforms.)

#### DOCUMENTATION UPDATE:

- The R Internals manual has been updated to better explain some aspects of pqR implementation.

#### FEATURE CHANGE:

- Parsed expressions no longer contain explicit parenthesis operators when the parentheses are necessary to override the precedence of operators. These necessary parentheses will be inserted when the expression is deparsed. See the help on `parse` and `deparse`. This change does impact a few packages (such as `coxme`) that consider the presence of parentheses in formulas to be significant. Formulas may be exempted from parenthesis suppression in a future release, but for now, such packages won't work.

#### PERFORMANCE IMPROVEMENTS:

- The overhead of interpreting R code has been reduced by various detailed code improvements, and by sometimes returning scalar integer and real values in special “static boxes”. As a result, the benefit of using the byte-code compiler is reduced. Note that in pqR using the byte-code compiler can often slow down functions, since byte-compiled code does not support some pqR optimizations such as task merging.
- Speed of evaluation for expressions with necessary parentheses will be faster because of the feature change mentioned above that eliminates them. Note that including unnecessary parentheses will still (slightly) slow down evaluation. (These unnecessary parentheses are preserved so that the expression will appear as written when deparsed.)
- Assignment to list elements, and other uses of the `$<-` operator, are now substantially faster.
- Coercion of logical and integer vectors to character vectors is now much faster, as is creation of names with sequence numbers.
- Operations that create strings are now sometimes faster, due to improvements in string hashing and memory allocation.

### PERFORMANCE IMPROVEMENTS FROM A LATER R CORE RELEASE:

- A number of performance improvements relating to S3 and S4 class implementation, due to Tomas Kalibera, were incorporated from R 3.2.0.

### BUG FIXES:

- A large number of fixes were made to correct reliability problems (mostly regarding protection of pointers). Many of these were provided by Tomas Kalibera as fixes to R Core versions (sometimes with adaptation required for use in pqR). Some were fixed in pqR and reported to R Core. Others were for problems only existing in pqR.
- Fixed a bug in which pqR's optimization of updates such as `a<-a+1` could sometimes permit modification of a locked binding.
- Fixed related problems with `apply`, `lapply`, `vapply`, and `eapply`, that can show up when the value returned by the function being applied is itself a function. This problem also resulted in incorrect display of saved warning messages. The problems are also fixed in R-3.2.0, in a different way.
- The `gctorture` function now works as documented, forcing a FULL garbage collection on every allocation. This does make running with gctorture enabled even slower than before, when most garbage collections were partial, but is more likely to find problems.
- Fixed a bug in `nls` when the `algorithm="port"` option is used, which could result in a call of `nls` being terminated with a spurious error message. This bug is most likely to arise on a 64-bit big-endian platform, such as a 64-bit SPARC build, but will occur with small probability on most platforms. It is also present in R Core versions of R.
- Fixed a bug in `readBin` in which a crash could occur due to misaligned data accesses. This bug is also present in R Core versions of R.

### BUG FIX CORRESPONDING TO ONE IN A LATER R CORE RELEASE:

- Removed incorrect information from `help(call)`, as also done in R-3.0.2.

## CHANGES IN VERSION RELEASED 2014-11-16

### INTRODUCTION:

- This and the previous release of 2014-10-23 (which does not have a separate NEWS section) are minor updates to the release of 2014-09-30, with fixes for a few problems, and a few performance improvements. Packages installed for pqR-2014-09-30 or pqR-2014-10-23 do not need to be reinstalled for this release.

### INSTALLATION, BUILDING, AND TESTING:

- For Mac OS X, a change has been made to allow use of the Accelerate framework for the BLAS in OS X 10.10 (Yosemite), adapted from a patch by R Core.
- A new test (`var-lookup.R`) for correctness of local vs. global symbol bindings has been added, which is run with other tests done by "make check".

### DOCUMENTATION UPDATES:

- The documentation on "contexts" in the R Internals manual has been updated to reflect a change made in pqR-2014-09-30. (The internals manual has also been updated to reflect changes below.)

**PERFORMANCE IMPROVEMENTS:**

- The speed of `for` loops has been improved by not bothering to set the index variable again if it is still set to the old value in its binding cell.
- Evaluation of symbols is now a bit faster when the symbol has a binding in the local environment whose location is cached.
- Lookup of functions now often skips local environments that were previously found not to contain the symbol being looked up. In particular, this speeds up calls of base functions that are not already fast due to their being recognized as "special" symbols.
- The set of "special" symbols for which lookups in local environments is usually particularly fast now includes `.C`, `.Fortran`, `.Call`, `.External`, and `.Internal`.
- Adjusted a tuning parameter for `rowSums` and `rowMeans` to be more appropriate for the cache size in modern processors.

**PERFORMANCE IMPROVEMENT FROM A LATER R CORE RELEASE:**

- The faster C implementation of diagonal matrix creation with `diag` from R-3.0.0 has been adapted for pqR.

**BUG FIXES:**

- Fixed a number of places in the interpreter and base packages where objects were not properly protected against garbage collection (many involving use of the `install` function). Most of these problems are in R-2.15.0 or R-2.15.1, and probably also in later R Core releases.
- Fixed a bug in which subsetting a vector with a range created with the colon operator that consisted entirely of invalid indexes could cause a crash (eg, `c(1,2)[10:20]`).
- Fixed a bug (pqR issue #27) in which a user-defined replacement function might get an argument that is not marked as shared, which could cause anomalous behaviour in some circumstances.
- Fixed an issue with passing on variant return requests to function bodies (though it's hard to construct an example where this issue produces incorrect results).
- Fixed a bug in initialization of user-supplied random number generators, which occasionally showed up in package `rngwell19937`.
- (Actually fixed in pqR-2014-09-30 but omitted from NEWS.) Fixed problems with calls of `strncpy` that were described in PR #15990 at [r-project.org](http://r-project.org).

**CHANGES IN VERSION RELEASED 2014-09-30****INTRODUCTION:**

- This release contains several major performance improvements. Notably, lookup of variables will sometimes be much faster, variable updates like `v <- v + 1` will often not allocate any new space, assignments to parts of variables (eg, `a[i] <- 0`) is much faster in the interpreter (no change for byte-compiled code), external functions called with `.Call` or `.External` now get faster macro or inline versions of functions such as `CAR`, `LENGTH`, and `REAL`, and calling of external functions with `.C` and `.Fortran` is substantially faster, and can sometimes be done in a helper thread.
- Changes have been made to configuration options regarding use of BLAS routines for matrix multiplication, as described below. In part, these changes are intended to make the default be close to what R Core releases do (but without the unnecessary inefficiency).

- A number of updates from R Core releases after R-2.15.0 have been incorporated or adapted for use in pqR. These provide some performance improvements, some new features or feature changes, and some bug fixes and documentation updates.
- Many other feature changes and performance improvements have also been made, as described below, and a number of bugs have been fixed, some of which are also present in the latest R Core release, R-3.1.1.
- Packages using `.Call` or `.External` should be re-installed for use with this version of pqR.

#### FEATURE CHANGES:

- The `mat_mult_with_BLAS` option, which controls whether the BLAS routines or pqR's C routines are used for matrix multiplication, may now be set to `NA`, which is equivalent to `FALSE`, except that for multiplication of sufficiently large matrices (not vector-vector, vector-matrix, or matrix-vector multiplication) pqR will use a BLAS routine unless there is an element in one of the operands that is `NA` or `NaN`. This mimics the behaviour of R Core implementations (at least through 3.1.1), which is motivated by a desire to ensure that `NA` is propagated correctly even if the BLAS does not do so, but avoids the substantial but needless inefficiency present in the R Core implementation.
- A `BLAS_in_helpers` option now allows run-time control of whether BLAS routines may be done in a helper thread. (But this will be fixed at `FALSE` if that is set as the default when pqR is built.)
- A `codePromises` option has been added to `deparse`, and documented in `help(.deparseOpts)`. With this option, the deparsed expression uses the code part of a promise, not the value, similarly to the existing `delayPromises` option, but without the extra text that that option produces.
- This new `codePromises` `deparse` option is now used when producing error messages and traceback output. This improves error messages in the new scheme for subset assignments (see the section on performance improvements below), and also avoids the voluminous output previously produced in circumstances such as the following:

```
`f<-` <- function (x,value) x[1,1] <- value
a <- 1
f(a) <- rep(123,1000) # gives an error
traceback()
```

This previously produced output with 1000 repetitions of 123 in the traceback produced following the error message. The traceback now instead shows the expression `rep(123,1000)`.

- The `evaluate` option for `dump` has been extended to allow access to the new `codePromises` `deparse` option. See `help(dump)`.
- The formal arguments of primitive functions will now be returned by `formals`, as they are shown when printed or with `args`. In R Core releases (at least to R-3.1.1), the result of `formals` for a primitive is `NULL`.
- Setting the `deparse.max.lines` option will now limit the number of lines printed when exiting debug of a function, as well as when entering.
- In `.C` and `.Fortran`, arguments may be character strings even when `DUP=FALSE` is specified - they are duplicated regardless. This differs from R Core versions, which (at least through R-3.1.1) give an error if an argument is a character string and `DUP=FALSE`.

- In `.C` and `.Fortran`, scalars (vectors of length one) are duplicated (in effect, though not necessarily physically) even when `DUP=FALSE` is specified. However, they are not duplicated in R Core versions (at least through R-3.1.1), so it may be unwise to rely on this.
- A `HELPER` argument can now be used in `.C` and `.Fortran` to specify that the C or Fortran routine may (sometimes) be done in a helper thread. (See the section on performance improvements below.)

#### FEATURE CHANGES CORRESPONDING TO THOSE IN LATER R CORE RELEASES:

- From R-3.0.2: The unary `+` operator now converts a logical vector to an integer vector.
- From R-3.0.0: Support for "converters" for use with `.C` has been dropped.
- From R-2.15.1: `pmin()` and `pmax()` now also work when one of the inputs is of length zero and others are not, returning a zero-length vector, analogously to, say, `+`.
- From R-2.15.1: `.C()` gains some protection against the misuse of character vector arguments. (An all too common error is to pass `character(N)`, which initializes the elements to `""`, and then attempt to edit the strings in-place, sometimes forgetting to terminate them.)
- From R-2.15.1: Calls to the new function `globalVariables()` in package `utils` declare that functions and other objects in a package should be treated as globally defined, so that CMD check will not note them.
- From R-2.15.1: `print(packageDescription(*))` trims the `Collate` field by default.
- From R-2.15.1: A new option `"show.error.locations"` has been added. When set to `TRUE`, error messages will contain the location of the most recent call containing source reference information. (Other values are supported as well; see `?options`.)
- From R-2.15.1: C entry points `R_GetCurrentSrcref` and `R_GetSrcFilename` have been added to the API to allow debuggers access to the source references on the stack.

#### INSTALLATION, BUILDING, TESTING, AND DEBUGGING:

- The `--enable-mat-mult-with-BLAS` configuration option has been replaced by the ability to use a `configure` argument of `mat_mult_in_BLAS=FALSE`, `mat_mult_in_BLAS=FALSE`, or `mat_mult_in_BLAS=NA`, to set the default value of this option.
- The `--disable-mat-mult-with-BLAS-in-helpers` configuration option has been replaced by the ability to use a `configure` argument of `BLAS_in_helpers=FALSE` or `BLAS_in_helpers=TRUE` to set the default value of this option.
- The LAPACK routines used are now the same as those in R-3.1.1 (version 3.5.0). However, the `.Call` interface to these remains as in R-2.15.0 to R-2.15.3 (it was changed to use `.Internal` in R-3.0.0). Since LAPACK 3.5.0 uses some more recent Fortran features, a Fortran 77 compiler such as `g77` will no longer suffice.
- Setting the environment variable `R_ABORT` to any non-null string will prevent any attempt to produce a stack trace on a segmentation fault, in favour of instead producing (maybe) an immediate core dump.
- The variable `R_BIT_BUCKET` in `'share/make/vars.mk'` now specifies a file to receive output that is normally ignored when building `pqR`. It is set to `'dev/null'` in the distribution, but this can be changed to help diagnose build problems.

- The C functions `R_inspect` and `R_inspect3` functions are now visible to package code, so they can be used there for debugging. To see what they do, look in `'src/main/inspect.c'`. They are subject to change, and should not appear in any code released to users.
- The `Rf_error` and related procedures declared in `'R_ext/Error.h'` are now if possible declared to never return, allowing for slightly better code generation by the compiler, and avoiding spurious compiler warnings. This parallels a change in R-3.0.1, but is more general, using the C11 noreturn facility if present, and otherwise resorting to the gcc facility (if gcc is used).

### INSTALLATION FEATURES LIKE THOSE IN LATER R CORE RELEASES:

- From R-2.15.1: `install.packages("pkg_version.tgz")` on Mac OS X now has sanity checks that this is actually a binary package (as people have tried it with incorrectly named source packages).
- From R-2.15.2: `--with-blas='-framework vecLib'` now also works on OS X 10.8 and 10.9.
- From R-2.15.3: Configuration and R CMD javareconf now come up with a smaller set of library paths for Java on Oracle-format JDK (including OpenJDK). This helps avoid conflicts between libraries (such as libjpeg) supplied in the JDK and system libraries. This can always be overridden if needed: see the 'R Installation and Administration' manual.
- From R-2.15.3: The configure tests for Objective C and Objective C++ now work on Mac OS 10.8 with Xcode 4.5.2 (PR#15107).
- The cairo-based versions of `X11()` now work with current versions of `cairographics` (e.g. 1.12.10). (PR#15168)

**DOCUMENTATION UPDATES:** These are in addition to changes in documentation relating to other changes reported here.

- Some incorrect code has been corrected in the "Writing R Extensions" manual, in the "Zero finding" and "Calculating numerical derivatives" sections. The discussion in "Finding and Setting Variables" has also been clarified to reflect current behaviour.
- Documentation in the "R Internals" manual has been updated to reflect recent changes in pqR regarding symbols and variable lookup, and to remove incorrect information about the global cache present in the version from R-2.15.0 (and R-3.1.1).
- Fixed an out-of-date comment in the section on helper threads in the "R Internals" manual.

**PERFORMANCE IMPROVEMENTS:** Numerous improvements in speed and memory usage have been made in this release of pqR. Some of these are noted here.

- Lookup of local variables is now usually much faster (especially when the number of local variables is large), since for each symbol, the last local binding found is now recorded, usually avoiding a linear search through local symbol bindings. Those lookups that are still needed are also now a bit faster, due to unrolling of the search loop.

- Assignments to selected parts of variables (eg, `a[i,j] <- 0` or `names(L$a[[f()]]) <- v`) are now much faster in the interpreter. (Such assignments within functions that are byte-compiled use a different mechanism that has not been changed in this release.)

This change also alters the error messages produced from such assignments. They are probably not as informative (at least to unsophisticated users) as those that the interpreter produced previously, though they are better than those produced from byte-compiled code. On the plus side, the error messages are now consistent for primitive and user-written replacement functions, and some messages now contain short, intelligible expressions that could previously contain huge amounts of data (see the section on new features above).

This change also fixes the anomaly that arguments of subset expressions would sometimes be evaluated more than once (eg, `f()` in the example above).

- The speed of `.C` and `.Fortran` has been substantially improved, partly by incorporating changes in R-2.15.1 and R-2.15.2, but with substantial additional improvements as well.
- The speed of `.Call` and `.External` has been improved somewhat. More importantly, the C routines called will get macro versions of `CAR`, `CDR`, `CADR`, etc., macro versions of `TYPEOF` and `LENGTH`, and inline function versions of `INTEGER`, `LOGICAL`, `REAL`, `COMPLEX`, and `RAW`. This avoidance of procedure call overhead for these operations may speed up some C procedures substantially.
- In some circumstances, a routine called with `.C` or `.Fortran` can now be done in a helper thread, in parallel with other computations. This is done only if requested with the `HELPER` option, and at present only in certain limited circumstances, in which only a single output variable is used. See `help(.C)` or `help(.Fortran)` for details.
- As an initial use of the previous feature, the `findInterval` function now will sometimes execute its C routine in a helper thread. (More significant uses of the `HELPER` option to `.C` and `.Fortran` will follow in later releases.)
- Assignments that update a local variable by applying a single unary or binary mathematical operation will now often re-use space for the variable that is updated, rather than allocating new space. For example, this will be done with all the assignments in the second line below:

```
u <- rep(1,1000); v <- rep(2,1000); w <- exp(2)
u <- exp(u); u <- 2*u; v <- v/2; u <- u+v; w <- w+1
```

This modification also has the effect of increasing the possibilities for task merging. For example, in the above code, the first two updates for `u` will be merged into one computation that sets `u` to `2*exp(u)` using a single loop over the vector.

- The performance of `rep` and `rep.int` is much improved. These improvements (and improvements previously made in `pqR`) go beyond those in R Core releases from R-2.15.2 on, so these functions are often substantially faster in `pqR` than in R-2.15.2 or later R Core versions to at least R-3.1.1, for both long and short vectors. (However, note that the changes in functionality made in R-2.15.2 have not been made in `pqR`; in particular, pairlists are still allowed, as in R-2.15.0.)
- For numeric vectors, the repetition done by `rep` and `rep.int` may now be done in a helper thread, in parallel with other computations. For example, attaching names to the result of `rep` (if necessary) may be done in parallel with replication of the data part.

- The amount of space used on the C stack has been reduced, with the result that deeper recursion is possible within a given C stack limit. For example, the following is now possible with the default stack limit (at least on one Intel Linux system with gcc 4.6.3, results will vary with platform):

```
f <- function (n) { if (n>0) 1+f(n-1) else 0 }
options(expressions=500000)
f(7000)
```

For comparison, with pqR-2014-06-1, and R-3.1.1, trying to evaluate `f(3100)` gives a C stack overflow error (but `f(3000)` works).

- Expressions now sometimes return constant values, that are shared, and likely stored in read-only memory. These constants include `NULL`, the scalars (vectors of length one) `FALSE`, `TRUE`, `NA`, `NA_real_`, `0.0`, `1.0`, `0L`, `1L`, ..., `10L`, and some one-element pairlists with such constant elements. Apart from `NULL`, these constants are not *always* used for the corresponding value, but they often are, which saves on memory and associated garbage collection time. External routines that incorrectly modify objects without checking that `NAMED` is zero may now crash when passed a read-only constant, which is a generally desirable debugging aid, though it might sometimes cause a package that had previously at least sort-of worked to no longer run.
- The `substr` function has been sped up, and uses less memory, especially when a small substring is extracted from a long string. Assignment to `substr` has also been sped up a bit.
- The function for unserializing data (eg, reading file `‘.RData’`) is now done with elimination of tail-recursion (on the `CDR` field) when reading pairlists. This is both faster and less likely to produce a stack overflow. Some other improvements to serializing/unserializing have also been made, including support for restoring constant values (mentioned above) as constant values.
- Lookup of `S3` methods has been sped up, especially when no method is found. This is important for several primitive functions, such as `$`, that look for a method when applied to an object with a class attribute, but perform the operation themselves if no method is found.
- Integer plus, minus, and times are now somewhat faster (a side effect of switching to a more robust overflow check, as described below).
- Several improvements relating to garbage collection have been made. One change is that the amount of memory used for each additional symbol has been reduced from 112 bytes (two `CONS` cells) to 80 bytes (on 64-bit platforms), not counting the space for the symbol’s name (a minimum of 48 bytes on 64-bit platforms). Another change is in tuning of heap sizes, in order to reduce occasions in which garbage collection is very frequent.
- Many uses of the `return` statement have been sped up.
- Functions in the `apply` family have been sped up when they are called with no additional arguments for the function being applied.
- The performance problem reported in PR #15798 at [r-project.org](http://r-project.org) has been fixed (differently from the R Core fix).
- A performance bug has been fixed in which any assignment to a vector subscripted with a string index caused the entire vector to be copied. For example, the final assignment in the code below would copy all of `a`:

```
a<-rep(1.1,10000); names(a)[1] <- "x"
```



```
a["x"] <- 9
```

This bug exists in R Core implementations though at least R-3.1.1.

- A performance bug has been fixed that involved subscripting with many invalid string indexes, reported on r-devel on 2010-07-15 and 2013-05-8. It is illustrated by the following code, which was more than ten thousand times slower than expected:

```
x <- c(A=10L, B=20L, C=30L)
subscript <- c(LETTERS[1:3], sprintf("ID%05d", 1:150000))
system.time(y1 <- x[subscript])
```

The fix in this version of pqR does not solve the related problem when assigning to `x[subscript]`, which is still slow. Fixing that would require implementation of a new method, possibly requiring more memory.

This performance bug exists in R Core releases through R-3.1.1, but may now be fixed (differently) in the current R Core development version.

#### BUG FIXES:

- Fixed a bug in `numericDeriv` (see also the documentation update above), which is illustrated by the following code, which gave the wrong derivative:

```
x <- y <- 10
numericDeriv(quote(x+y), c("x", "y"))
```

I reported this to R Core, and it is also fixed (differently) in R-3.1.1.

- Fixed a problem in `.C` and `.Fortran` where, contrary to the documentation (except when `DUP=TRUE` and no duplication was actually needed), logical values after the call other than `TRUE`, `FALSE`, and `NA` are not mapped to `TRUE`, but instead exist as invalid values that may show up later. This bug exists in R Core versions 2.15.1 through at least 3.1.1. I reported it as [PR#15878](#) at r-project.org, so it may be fixed in a later R Core release.
- Fixed a problem with treatment of `ANYXP` in specifying types of registered C or Fortran routines, which in particular had prevented the types of `str_signif`, used in `formatC`, from being registered. (This bug exists in R Core versions of R at least through R-3.1.1.)
- Fixed a bug in `substr` applied to a string with UTF-8 encoding, which could cause a crash for code such as

```
a <- "\xc3\xa9"
Encoding(a) <- "UTF-8"
b <- paste0(rep(a, 8000), collapse="")
c <- substr(b, 1, 16000)
```

I reported this as [PR15910](#) at r-project.org, so it may be fixed in an R Core release after R-3.1.1. A related bug in assignment to `substr` has also been fixed.

- Fixed a bug in how debugging is handled that is illustrated by the following output:

```
> g <- function () { A <- A+1; function (x) x+1 }
> f <- function () (g())(10)
> A <- 0; f(); print(A)
[1] 11
[1] 1
> debug(f);
> A <- 0; f(); print(A)
debugging in: f()
```

```

debug: (g())(10)
Browse[2]> c
exiting from: f()
[1] 11
[1] 2

```

Note that the final value of **A** is different (and wrong) when **f** is stepped through in the debugger. This bug exists in R Core releases through at least R-3.1.1.

- Fixed a bug illustrated by the following code, which gave an error saying that **p[1,1]** has the wrong number of subscripts:

```
p <- pairlist(1,2,3,4); dim(p) <- c(2,2); p[1,1] <- 9
```

This bug exists in R Core releases through at least R-3.1.1.

- Fixed the following pqR bug (and related bugs), in which **b** was modified by the assignment to **a**:

```

a <- list(list(1+1))
b <- a
attr(a[[1]][[1]], "fred") <- 9
print(b)

```

- Fixed the following bug in which **b** was modified by an assignment to **a** with a vector subscript:

```

a <- list(list(mk2(1)))
b <- a[[1]]
a[[c(1,1)]] [1] <- 3
print(b)

```

This bug also exists in R-2.15.0, but was fixed in R-3.1.1 (quite differently than in pqR).

- Fixed a lack of error checking bug that could cause expressions such as `match.call(), expression()` to crash from an invalid memory reference. This bug also exists in R-2.15.0 and R-3.1.1.
- Fixed the non-robust checks for integer overflow, which reportedly sometimes fail when using clang on a Mac. This is #PR 15774 at r-project.org, fixed in R-3.1.1, but fixed differently in pqR.
- Fixed a pqR bug with expressions of the form `t(x)%*%y` when **y** is an S4 object.
- Fixed a bug (PR #15399 at r-project.org) in `na.omit` and `na.exclude` that led to a data frame that should have had zero rows having one row instead. (Also fixed in R-3.1.1, though differently.)
- Fixed the problem that RStudio crashed whenever a function was debugged (with `debug`). This was due to pqR having changed the order of fields in the `RCNTEXT` structure, which is an internal data structure of the interpreter, but is nevertheless accessed in RStudio. The order of fields is now back to what it was.
- Fixed the bug in `nlm` reported as PR #15958 at r-project.org, along with related bugs in `uniroot` and `optimize`. These all involve situations where the function being optimized saves its argument in some manner, and then sees the saved value change when the optimizer re-uses the space for the argument on the next call. The fix made is to no longer reuse the space, which will unfortunately cause a (fairly small) decline in performance.

The `optim` function also has this problem, but only when numerical derivatives are used. It has not yet been fixed. The `integrate` function does not seem to have a problem.

- Fixed a bug in the code to check for C stack overflow, that may show up when the fallback method for determining the start of the stack is needed, and a stack check is then done when very little stack is in use, resulting in an erroneous report of stack overflow. The problem is platform dependent, but arises on a SPARC Solaris system when using gcc 3.4.3, once stack usage is reduced by the improvement described above, leading to failure of one of the tests for package Matrix. This bug exists in R Core version back to 2.11.1 (or earlier) and up to at least 3.1.1.

#### **BUG FIXES CORRESPONDING TO THOSE IN LATER R CORE RELEASES:**

- From R-2.15.1: Trying to update (recommended) packages in `R_HOME/library` without write access is now dealt with more gracefully. Further, such package updates may be skipped (with a warning), when a newer installed version is already going to be used from `.libPaths()`. (PR#14866)
- From R-2.15.1: R CMD check with `_R_CHECK_NO_RECOMMENDED_` set to a true value (as done by the `--as-cran` option) could issue false errors if there was an indirect dependency on a recommended package.
- From R-2.15.1: `getMethod(f, sig)` produced an incorrect error message in some cases when `f` was not a string).
- From R-2.15.2: In Windows, the GUI preferences for foreground color were not always respected. (Reported by Benjamin Wells.)
- From R-2.15.1: The evaluator now keeps track of source references outside of functions, e.g. when `source()` executes a script.
- From R-2.15.1: The value returned by `tools::psnice()` for invalid pid values was not always NA as documented.
- From R-2.15.2: `sort.list(method = "radix")` could give incorrect results on certain compilers (seen with `clang` on Mac OS 10.7 and `Xcode 4.4.1`).
- From R-3.0.1: Calling `file.copy()` or `dirname()` with the invalid input `""` (which was being used in packages, despite not being a file path) could have caused a segfault.
- From R-3.0.1: `dirname("")` is now `""` rather than `."` (unless it segfaulted).
- Similarly to R-3.1.1-patched: In package `parallel`, child processes now call `_Exit` rather than `exit`, so that the main process is not affected by flushing of input/output buffers in the child.

#### **CHANGES IN VERSION RELEASED 2014-06-19**

##### **INTRODUCTION:**

- This is a maintenance release, with bug fixes, documentation improvements (including provision of previously missing documentation), and changes for compatibility with R Core releases. There are some new features in this release that help with testing pqR and packages. There are no significant changes in performance.
- See the sections below on earlier releases for general information on pqR.
- Note that there was a test release of 2014-06-10 that is superceded by this release, with no separate listing of the changes it contained.

**NEW FEATURES FOR TESTING:**

- The setting of the `R_SEED` environment variable now specifies what random number seed to use when `set.seed` is not called. When `R_SEED` is not set, the seed will be set from the time and process ID as before. It is recommended that `R_SEED` be set before running tests on pqR or packages, so that the results will be reproducible. For example, some packages report an error if a hypothesis test on simulated data results in a p-value less than some threshold. If `R_SEED` is not set, these packages will fail their tests now and then at random, whereas setting `R_SEED` will result either in consistent success or (less likely) consistent failure.
- The comparison of test output with saved output using `Rdiff` now ignores any output from `valgrind`, so spurious errors will not be triggered by using it. When using `valgrind`, the output files should be checked manually for `valgrind` messages that are of possible interest.
- The test script in `'tests/internet.R'` no longer looks at CRAN's html code, which is subject to change. It instead looks at a special test file at [pqR-project.org](http://pqR-project.org).
- Fixed problems with the `'reg-tests-1b'` test script. Also, now sets the random seed, so it's consistent (even without `R_SEED` set), and has its output compared to a saved version. Non-fatal errors (with code 5) should be expected on systems without enough memory for xz compression.

**CHANGE FOR COMPATIBILITY:**

- The result of `diag(list(1,3,5))` is now a matrix of type double. In R-2.15.0, this expression did not produce a sensible result. A previous fix in pqR made this expression produce a matrix of type list. A later change by R Core also fixed this, but so it produced a double matrix, coercing the list to a numeric vector (to the extent possible); pqR now does the same.

**DOCUMENTATION UPDATES:**

- The documentation for `c` now says how the names for the result are determined, including previously missing information on the `use.names` argument, and on the role of the names of arguments in the call of `c`. This documentation is missing in R-2.15.0 and R-3.1.0.
- The documentation for `diag` now documents that a diagonal matrix is always created with type double or complex, and that the names of an extracted diagonal vector are taken from a `names` attribute (if present), if not from the row and column names. This information is absent in the documentation in R-2.15.1 and R-3.1.0.
- Incorrect information regarding the pointer protection stack was removed from `help(Memory)`. This incorrect information is present in R-2.15.0 and R-3.1.0 as well.
- There is now information in `help(Arithmetic)` regarding what happens when the operands of an arithmetic operation are NA or NaN, including the arbitrary nature of the result when one operand is NA and the other is NaN. There is no discussion of this issue in the documentation for R-2.15.0 and R-3.1.0.
- The `R_HELPERS` and `R_HELPERS_TRACE` environment variables are now documented in `help("environment variables")`. The documentation in `help(helpers)` has also been clarified.
- The `R_DEBUGGER` and `R_DEBUGGER_ARGS` environment variables are now documented in `help("environment variables")` as alternatives to the `--debugger` and `--debugger-args` arguments.

**BUG FIXES:**

- Fixed lack of protection bugs in the `equal` and `greater` functions in `'sort.c'`. These bugs are also present in R-2.15.0 and R-3.1.0.
- Fixed lack of protection bugs in the `D` function in `'deriv.c'`. These bugs are also present in R-2.15.0 and R-3.1.0.
- Fixed argument error-checking bugs in `getGraphicsEventEnv` and `setGraphicsEventEnv` (also present in R-2.15.0 and R-3.1.0).
- Fixed a stack imbalance bug that shows up in the expression `anyDuplicated(c(1,2,1),incomp=2)`. This bug is also present in R-2.15.0 and R-3.1.0. The bug is reported only when the `base` package is not byte compiled (but still exists silently when it is compiled).
- Fixed a bug in the foreign package that showed up on systems where the C `char` type is unsigned, such as a Raspberry Pi running Raspbian. I reported this to R Core, and it is also fixed in R-3.1.0.
- Fixed a lack of protection bug that arose when `log` produced a warning.
- Fixed a lack of protection bug in the `lang[23456]` C functions.
- Fixed a stack imbalance bug that showed up when an assignment was made to an array of three or more dimensions using a zero-length subscript.
- Fixed a problem with `news()` that was due to pqR's version numbers being dates (pqR issue #1).
- Fixed out-of-bound memory accesses in `R_chull` and `scanFrame` that valgrind reports (but which are likely to be innocuous).

**BUG FIXES CORRESPONDING TO THOSE IN LATER R CORE RELEASES:**

- From R-2.15.1: The string "infinity" now converts correctly to `Inf` (PR#14933).
- From R-2.15.1: The generic for `backsolve` is now correct (PR#14883).
- From R-2.15.1: A bug in `get_all_vars` was fixed (PR#14847).
- From R-2.15.1: Fixed an argument error checking bug in `dev.set`.
- From R-3.1.0-patched: Fixed a problem with `mcmapply` not parallelizing when the number of jobs was less than number of cores. (However, unlike R-3.1.0-patched, this fix doesn't try to parallelize when there is only one core.)

**CHANGES IN VERSION RELEASED 2014-02-23****INTRODUCTION:**

- This is a maintenance release, with bug fixes, changes for compatibility with packages, additional correctness tests, and documentation improvements. There are no new features in this release, and no significant changes in performance.
- See the sections below on earlier releases for general information on pqR.

**INSTALLATION AND TESTING:**

- The information in the file "INSTALL" in the main source directory has been re-written. It now contains all the information expected to be needed for most installations, without the user needing to refer to R-admin, including information on the configuration options that have been added for pqR. It also has information on how to build pqR from a development version downloaded from github.

- Additional tests regarding subsetting operations, maintenance of NAMEDCNT, and operation of helper threads have been written. They are run with `make check` or `make check-all`.
- A "create-configure" shell script is now included, which allows for creation of the "configure" shell script when it is non-functional or not present (as when building from a development version of pqR). It is not needed for typical installs of pqR releases.
- Some problems with installation on Microsoft Windows (identified by Yu Gong) have hopefully been fixed. (But trying to install pqR on Windows is still recommended only for adventurous users.)
- A problem with installing pqR as a shared library when multithreading is disabled has been fixed.
- Note that any packages (except those written only in R, plus C or Fortran routines called by `.C` or `.Fortran`) that were compiled and installed under R Core versions of R must be re-installed for use with pqR, as is generally the case with new versions of R (although it so happens that it is not necessary to re-install packages installed with pqR-2013-07-22 or pqR-2013-12-29 with this release, because the formats of the crucial internal data structures happen not to have changed).

#### DOCUMENTATION UPDATES:

- The instructions in "INSTALL" have been re-written, as noted above.
- The manual on "Writing R Extensions" now has additional information (in the section on "Named objects and copying") on paying proper attention to NAMED for objects found in lists.
- More instructions on how to create a release branch of pqR from a development branch have been added to `mods/README` (or `MODS`).

#### CHANGES REGARDING PACKAGE COMPATIBILITY AND CHECKING:

- Changed the behaviour of `$` when dispatching so that the unevaluated element name arrives as a string, as in R-2.15.0. This behaviour is needed for the "dyn" package. The issue is illustrated by the following code:

```
a <- list(p=3,q=4)
class(a) <- "fred"
`$.fred` <-
  function (x,n) { print(list(n,substitute(n))); x[[n]] }
print(a$q)
```

In R-2.15.0, both elements of the list printed are strings, but in pqR-2013-12-29, the element from "substitute" is a symbol. Changed `help("$")` to document this behaviour, and the corresponding behaviour of `"$<-"`. Added a test with `make check` for it.

- Redefined "fork" to "Rf\_fork" so that helper threads can be disabled in the child when "fork" is used in packages like "multicore". (Special mods for this had previously been made to the "parallel" package, but this is a more universal scheme.)
- Added an option (currently set) for pqR to ignore incorrect zero pointers encountered by the garbage collector (as R-2.15.0 does). This avoids crashes with some packages (eg, "birch") that incorrectly set up objects with zero pointers.

- Changed a C procedure name in the "matprod" routines to reduce the chance of a name conflict with C code in packages.
- Made `NA_LOGICAL` and `NA_INTEGER` appear as variables (rather than constants) in packages, as needed for package "RcppEigen".
- Made `R_CStackStart` and `R_CStackLimit` visible to packages, as needed for package "vimcom".
- Fixed problem with using `NAMED` in a package that defines `USE_RINTERNALS`, such as "igraph".
- Calls of external routines with `.Call` and `.External` are now followed by checks that the routine didn't incorrectly change the constant objects sometimes used internally in pqR for `TRUE`, `FALSE`, and `NA`. (Previously, such checks were made only after calls of `.C` and `.Fortran`.)

#### BUG FIXES:

- Fixed the following bug (also present in R-2.15.0 and R-3.0.2):

```
x <- t(5)
print (x %*% c(3,4))
print (crossprod(5,c(3,4)))
```

The call of `crossprod` produced an error, whereas the corresponding use of `%*%` does not.

In pqR-2013-12-29, this bug also affected the expression `t(5) %*% c(3,4)`, since it is converted to the equivalent of `crossprod(5,c(3,4))`.

- Fixed a problem in `R_AllocStringBuffer` that could result in a crash due to an invalid memory access. (This bug is also present in R-2.15.0 and R-3.0.2.)
- Fixed a bug in a "matprod" routine sometimes affecting `tcrossprod` (or an equivalent use of `%*%`) with helper threads.
- Fixed a bug illustrated by the following:

```
f <- function (a)
{ x <- a
  function () { b <- a; b[2]<-1000; a+b }
}
g <- f(c(7,8,9))
save.image("tmpimage")
load("tmpimage")
print(g())
```

where the result printed was 14 2000 18 rather than 14 1008 18.

- Fixed a bug in `prod` with an integer vector containing `NA`, such as, `prod(NA)`.
- Fixed a lack-of-protection bug in `mkCharLenCE` that showed up in checks for packages "cmrutils".
- Fixed a problem with `xtfrm` demonstrated by the following:

```
f<-function(...) xtfrm(...); f(c(1,3,2))
```

 which produced an error saying `'...'` was used in an incorrect context. This affected package "lsr".
- Fixed a bug in maintaining `NAMEDCNT` when assigning to a variable in an environment using `$`, which showed up in package "plus".
- Fixed a bug that causes the code below to create a circular data structure:

```
{ a <- list(1); a[[1]] <- a; a }
```

- Fixed bugs such as that illustrated below:

```
a <- list(list(list(1)))
b <- a
a[[1]][[1]][[1]]<-2
print(b)
```

in which the assignment to `a` changes `b`, and added tests for such bugs.

- Fixed a bug where unary minus might improperly reuse its operand for the result even when it was logical (eg, in `-c(F,T,T,F)`).
- Fixed a bug in `pairlist` element deletion, and added tests in `subset.R` for such cases.
- The ISNAN trick (if enabled) is now used only in the interpreter itself, not in packages, since the macro implementing it evaluates its argument twice, which doesn't work if it has side effects (as happens in the "ff" package).
- Fixed a bug that sometimes resulted in task merging being disabled when it shouldn't have been.

## CHANGES IN VERSION RELEASED 2013-12-29

### INTRODUCTION:

- This is the first publicized release of `pqR` after `pqR-2013-07-22`. A version dated 2013-11-28 was released for testing; it differs from this release only in bug and documentation fixes, which are not separately detailed in this NEWS file.
- `pqR` is based on R-2.15.0, distributed by the R Core Team, but improves on it in many ways, mostly ways that speed it up, but also by implementing some new features and fixing some bugs. See the notes below on earlier `pqR` releases for general discussion of `pqR`, and for information that has not changed from previous releases of `pqR`.
- The most notable change in this release is that "task merging" is now implemented. This can speed up sequences of vector operations by merging several operations into one, which reduces time spent writing and later reading data in memory. See `help(merging)` and the item below for more details.
- This release also includes other performance improvements, bug fixes, and code cleanups, as detailed below.

### INSTALLATION AND TESTING:

- Additional configuration options are now present to allow enabling and disabling of task merging, and more generally, of the deferred evaluation framework needed for both task merging and use of helper threads. By default, these facilities are enabled. The `--disable-task-merging` option to `./configure` disables task merging, `--disable-helper-threads` disables support for helper threads (as before), and `--disable-deferred-evaluation` disables both of these features, along with the whole deferred evaluation framework. See the `R-admin` manual for more details.
- See the `pqR` wiki at <https://github.com/radfordneal/pqR/wiki> for the latest news regarding systems and packages that do or do not work with `pqR`.
- Note that any packages (except those written only in R, plus C or Fortran routines called by `.C` or `.Fortran`) that were compiled and installed under R Core versions of R must be re-installed for use with `pqR`, as is generally the case with new versions of R (although it so happens that it is not necessary to re-install packages installed



with pqR-2013-07-22 with this release, because the formats of the crucial internal data structures happen not to have changed).

- Additional tests of matrix multiplication (`%*%`, `crossprod`, and `tcrossprod`) have been written. They are run with `make check` or `make check-all`.

## INTERNAL STRUCTURES AND APPLICATION PROGRAM INTERFACE:

- The table of built-in function names, C functions implementing them, and operation flags, which was previously found in `src/main/names.c`, has been split into multiple tables, located in the source files that define such built-in functions (with only a few entries still in `names.c`). This puts the descriptions of these built-in functions next to their definitions, improving maintainability, and also reduces the number of global functions. This change should have no effects visible to users.
- The initialization for fast dispatch to some primitive functions is now done in `names.c`, using tables in other source files analogous to those described in the point just above. This is cleaner, and eliminates an anomaly in the previous versions of pqR that a primitive function could be slower the first time it was used than when used later.

## PERFORMANCE IMPROVEMENTS:

- Some sequences of vector operations can now be merged into a single operation, which can speed them up by eliminating memory operations to store and fetch intermediate results. For example, when `v` is a long vector, the expression `exp(v+1)` can be merged into one task, which will compute `exp(v[i]+1)` for each element, `i`, of `v` in a single loop.

Currently, such “task merging” is done only for (some) operations in which only one operand is a vector. When there are helper threads (which might be able to do some operations even faster, in parallel) merging is done only when one of the operations merged is a simple addition, subtraction, or multiplication (with one vector operand and one scalar operand).

See `help(merging)` for more details.

- During all garbage collections, any tasks whose outputs are not referenced are now waited for, to allow memory used by their outputs to be recovered. (Such unreferenced outputs should be rare in real programs.) In a full garbage collection, tasks with large inputs or outputs that are referenced only as task inputs are also waited for, so that the memory they occupy can be recovered.
- The built-in C matrix multiplication routines and those in the supplied BLAS have both been sped up, especially those used by `crossprod` and `tcrossprod`. This will of course have no effect if a different BLAS is used and the `mat_mult_with_BLAS` option is set to `TRUE`.
- Matrix multiplications in which one operand can be recognized as the result of a transpose operation are now done without actually creating the transpose as an intermediate result, thereby reducing both computation time and memory usage. Effectively, these uses of the `%*%` operator are converted to uses of `crossprod` or `tcrossprod`. See `help("%*%")` for details.
- Speed of `ifelse` has been improved (though it's now slower when the condition is scalar due to the bug fix mentioned below).
- Inputs to the mod operator can now be piped. (Previously, this was inadvertently prevented in some cases.)

- The speed of the quick check for NA/NaN that can be enabled with `-DENABLE_ISNAN_TRICK` in CFLAGS has been improved.

#### BUG FIXES:

- Fixed a bug in `ifelse` with scalar condition but other operands with length greater than one. (Pointed out by Luke Tierney.)
- Fixed a bug stemming from re-use of operand storage for a result (pointed out by Luke Tierney) illustrated by the following:

```
A <- array(c(1), dim = c(1,1), dimnames = list("a", 1))
x <- c(a=1)
A/(pi*x)
```

- The `--disable-mat-mult-with-BLAS-in-helpers` configuration setting is now respected for complex matrix multiplication (previously it had only disabled use of the BLAS in helper threads for real matrix multiplication).
- The documentation for `aperm` now says that the default method does not copy attributes (other than dimensions and dimnames). Previously, it incorrectly said it did (as is the case also in R-2.15.0 and R-3.0.2).
- Changed `apply` from previous versions of pqR to replicate the behaviour seen in R-2.15.0 (and later R Core version) when the matrix or array has a class attribute. Documented this behaviour (which is somewhat dubious and convoluted) in the help entry for `apply`. This change fixes a problem seen in package TSA (and probably others).
- Changed `rank` from previous versions of pqR to replicate the behaviour when it is applied to data frames that is seen in R-2.15.0 (and later R Core versions). Documented this (somewhat dubious) behaviour in the help entry for `rank`. This change fixes a problem in the `coin` package.
- Fixed a bug in keeping track of references when assigning repeated elements into a list array.
- Fixed the following bug (also present in R-2.15.0 and R-3.0.2):

```
v <- c(1,2)
m <- matrix(c(3,4),1,2)
print(t(m)%*%v)
print(crossprod(m,v))
```

in which `crossprod` gave an error rather than produce the answer for the corresponding use of `%*%`.

- Bypassed a problem with the Xcode gcc compiler for the Mac that led to it falsely saying that using `-DENABLE_ISNAN_TRICK` in CFLAGS doesn't work.

#### CHANGES IN VERSION RELEASED 2013-07-22

##### INTRODUCTION:

- pqR is based on R-2.15.0, distributed by the R Core Team, but improves on it in many ways, mostly ways that speed it up, but also by implementing some new features and fixing some bugs. See the notes below, on the release of 2013-06-28, for general discussion of pqR, and for information on pqR that has not changed since that release.
- This updated release of pqR provides some performance enhancements and bug fixes, including some from R Core releases after R-2.15.0. More work is still needed to incorporate improvements in R-2.15.1 and later R Core releases into pqR.

- This release is the same as the briefly-released version of 2013-17-19, except that it fixes one bug and one reversion of an optimization that were introduced in that release, and tweaks the Windows Makefiles (which are not yet fully tested).

#### FEATURE AND DOCUMENTATION CHANGES:

- Detailed information on what operations can be done in helper threads is now provided by `help(helpers)`.
- Assignment to parts of a vector via code such as `v[[i]]<-value` and `v[ix]<-values` now automatically converts raw values to the appropriate type for assignment into numeric or string vectors, and assignment of numeric or string values into a raw vector now results in the raw vector being first converted to the corresponding type. This is consistent with the existing behaviour with other types.
- The allowed values for assignment to an element of an "expression" list has been expanded to match the allowed values for ordinary lists. These values (such as function closures) could previously occur in expression lists as a result of other operations (such as creation with the `expression` primitive).
- Operations such as `v <- pairlist(1,2,3); v[[-2]] <- NULL` now raise an error. These operations were previously documented as being illegal, and they are illegal for ordinary lists. The proper way to do this deletion is `v <- pairlist(1,2,3); v[-2] <- NULL`.
- Raising `-Inf` to a large value (eg, `(-Inf)^(1e16)`) no longer produces an incomprehensible warning. As before, the value returned is `Inf`, because (due to their limited-precision floating-point representation) all such large numbers are even integers.

#### FEATURE CHANGES CORRESPONDING TO THOSE IN LATER R CORE RELEASES:

- From R-2.15.1: On Windows, there are two new environment variables which control the defaults for command-line options.  
If `R_WIN_INTERNET2` is set to a non-empty value, it is as if `'--internet2'` was used.  
If `R_MAX_MEM_SIZE` is set, it gives the default memory limit if `'--max-mem-size'` is not specified: invalid values being ignored.
- From R-2.15.1: The NA warning messages from e.g. `pchisq()` now report the call to the closure and not that of the `.Internal`.
- The following included software has been updated to new versions: `zlib` to 1.2.8, `LZMA` to 5.0.4, and `PCRE` to 8.33.

#### INSTALLATION AND TESTING:

- See the pqR wiki at <https://github.com/radfordneal/pqR/wiki> for the latest news regarding systems and packages that do or do not work with pqR.
- Note that any previously-installed packages must be re-installed for use with pqR (as is generally the case with new versions of R), except for those written purely in R.
- It is now known that pqR can be successfully installed under Mac OS X for use via the command line (at least with some versions of OS X). The gcc 4.2 compiler supplied by Apple with Xcode works when helper threads are disabled, but does not have the full OpenMP support required for helper threads. For helper threads to work, a

C compiler that fully supports OpenMP is needed, such as gcc 4.7.3 (available via [macports.org](http://macports.org)).

The Apple BLAS and LAPACK routines can be used by giving the `--with-blas='-framework vecLib'` and `--withlapack` options to `configure`. This speeds up some operations but slows down others.

The R Mac GUI would need to be recompiled for use with pqR. There are problems doing this unless helper threads are disabled (see pqR issue #17 for discussion).

Compiled binary versions of pqR for Mac OS X are not yet being supplied. Installation on a Mac is recommended only for users experienced in installation of R from source code.

- Success has also been reported in installing pqR on a Windows system, including with helper threads, but various tweaks were required. Some of these tweaks are incorporated in this release, but they are probably not sufficient for installation "out of the box". Attempting to install pqR on Windows is recommended only for users who are both experienced and adventurous.
- Compilation using the `-O3` option for gcc is not recommended. It speeds up some operations, but slows down others. With gcc 4.7.3 on a 32-bit Intel system running Ubuntu 13.04, compiling with `-O3` causes compiled functions to crash. (This is not a pqR issue, since the same thing happens when R-2.15.0 is compiled with `-O3`).

#### INTERNAL STRUCTURES AND APPLICATION PROGRAM INTERFACE:

- The R internals manual now documents (in Section 1.8) a preliminary set of conventions that pqR follows (not yet perfectly) regarding when objects may be modified, and how `NAMEDCNT` should be maintained. R-2.15.0 did not follow any clear conventions.
- The documentation in the R internals manual on how helper threads are implemented in pqR now has the correct title. (It would previously have been rather hard to notice.)

#### PERFORMANCE IMPROVEMENTS:

- Some unnecessary duplication of objects has been eliminated. Here are three examples: Creation of lists no longer duplicates all the elements put in the list, but instead increments `NAMEDCNT` for these elements, so that

```
a <- numeric(10000)
k <- list(1,a)
```

no longer duplicates `a` when `k` is created (though a duplication will be needed later if either `a` or `k[[2]]` is modified). Furthermore, the assignment below to `b$x`, no longer causes duplication of the 10000 elements of `y`:

```
a <- list (x=1, y=seq(0,1,length=10000))
b <- a
b$x <- 2
```

Instead, a single vector of 10000 elements is shared between `a$y` and `b$y`, and will be duplicated later only if necessary. Unnecessary duplication of a 10000-element vector is also avoided when `b[1]` is assigned to in the code below:

```
a <- list (x=1, y=seq(0,1,length=10000))
b <- a$y
a$y <- 0
b[1] <- 1
```

The assignment to `a$y` now reduces `NAMEDCNT` for the vector bound to `b`, allowing it to be changed without duplication.

- Assignment to part of a vector using code such as `v[101:200]<-0` will now not actually create a vector of 100 indexes, but will instead simply change the elements with indexes 101 to 200 without creating an index vector. This optimization has not yet been implemented for matrix or array indexing.
- Assignments to parts of vectors, matrices, and arrays using `"["` has been sped up by detailed code improvements, quite substantially in some cases.
- Subsetting of arrays of three or more dimensions using `"["` has been sped up by detailed code improvements.
- Pending summations of one-argument mathematical functions are now passed on by `sum`. So, for example, in `sum(exp(a)) + sum(exp(b))`, the two summations of exponentials can now potentially be done in parallel.
- A full garbage collection now does not wait for all tasks being done by helpers to complete. Instead, only tasks that are using or computing variables that are not otherwise referenced are waited for (so that this storage can be reclaimed).

#### BUG FIXES:

- A bug that could have affected the result of `sum(abs(v))` when it is done by a helper thread has been fixed.
- A bug that could have allowed `as.vector`, `as.integer`, etc. to pass on an object still being computed to a caller not expecting such a pending object has been fixed.
- Some bugs in which production of warnings at inopportune times could have caused serious problems have been fixed.
- The bug illustrated below (pqR issue #13) has been fixed:

```
> l = list(list(list(1)))
> l1 = l[[1]]
> l[[c(1,1,1)]] <- 2
> l1
[[1]]
[[1]][[1]]
[1] 2
```

- Fixed a bug (also present in R-2.15.0 and R-3.0.1) illustrated by the following code:

```
> a <- list(x=c(1,2),y=c(3,4))
> b <- as.pairlist(a)
> b$x[1] <- 9
> print(a)
$x
[1] 9 2

$y
[1] 3 4
```

The value printed for `a` has `a$x[1]` changed to 9, when it should still be 1. See pqR issue #14.

- Fixed a bug (also present in R-2.15.0 and R-3.0.1) in which extending an "expression" by assigning to a new element changes it to an ordinary list. See pqR issue #15.

- Fixed several bugs (also present in R-2.15.0 and R-3.0.1) illustrated by the code below (see pqR issue #16):

```
v <- c(10,20,30)
v[[2]] <- NULL          # wrong error message

x <- pairlist(list(1,2))
x[[c(1,2)]] <- NULL     # wrongly gives an error, referring to misuse
                        # of the internal SET_VECTOR_ELT procedure

v<-list(1)
v[[quote(abc)]] <- 2     # internal error, this time for SET_STRING_ELT

a <- pairlist(10,20,30,40,50,60)
dim(a) <- c(2,3)
dimnames(a) <- list(c("a","b"),c("x","y","z"))
print(a)                # doesn't print names

a[["a","x"]] <- 0        # crashes with a segmentation fault
```

#### BUG FIXES CORRESPONDING TO THOSE IN LATER R CORE RELEASES:

- From R-2.15.1: `formatC()` uses the C entry point `str_signif` which could write beyond the length allocated for the output string.
- From R-2.15.1: `plogis(x, lower = FALSE, log.p = TRUE)` no longer underflows early for large `x` (e.g. 800).
- From R-2.15.1: `?Arithmetic`'s " $1 \wedge y$  and  $y \wedge 0$  are 1, *always*" now also applies for integer vectors `y`.
- From R-2.15.1: X11-based pixmap devices like `png(type = "Xlib")` were trying to set the cursor style, which triggered some warnings and hangs.
- From R-3.0.1 patched: Fixed comment-out bug in BLAS, as per PR 14964.

#### CHANGES IN VERSION RELEASED 2013-06-28

##### INTRODUCTION:

- This release of pqR is based on R-2.15.0, distributed by the R Core Team, but improves on it in many ways, mostly ways that speed it up, but also by implementing some new features and fixing some bugs. One notable speed improvement in pqR is that for systems with multiple processors or processor cores, pqR is able to do some numeric computations in parallel with other operations of the interpreter, and with other numeric computations.
- This is the second publicised release of pqR (the first was on 2013-06-20, and there were earlier unpublicised releases). It fixes one significant pqR bug (that could cause two empty strings to not compare as equal, reported by Jon Clayden), fixes a bug reported to R Core (PR 15363) that also existed in pqR (see below), fixes a bug in deciding when matrix multiplies are best done in a helper thread, and fixes some issues preventing pqR from being built in some situations (including some partial fixes for Windows suggested by "armgong"). Since the rest of the news is almost unchanged

from the previous release, I have not made a separate news section for this release. (New sections will be created once new releases have significant differences.)

- This section documents changes in pqR from R-2.15.0 that are of direct interest to users. For changes from earlier version of R to R-2.15.0, see the `ONEWS`, `OONEWS`, and `OOONEWS` files. Changes of little interest to users, such as code cleanups and internal details on performance improvements, are documented in the file `MODS`, which relates these changes to branches in the code repository at [github.com/radfordneal/pqR](https://github.com/radfordneal/pqR).
- Note that for compatibility with R's version system, pqR presently uses the same version number, 2.15.0, as the version of R on which it is based. This allows checks for feature availability to continue to work. This scheme will likely change in the future. Releases of pqR with the same version number are distinguished by release date.
- See [radfordneal.github.io/pqR](https://radfordneal.github.io/pqR) for current information on pqR, including announcements of new releases, a link to the page for making and viewing reports of bugs and other issues, and a link to the wiki page containing information such as systems on which pqR has been tested.

#### FEATURE CHANGES:

- A new primitive function `get_rm` has been added, which removes a variable while returning the value it had when removed. See `help(get_rm)` for details, and how this can sometimes improve efficiency of R functions.
- An enhanced version of the `Rprofmem` function for profiling allocation of vectors has been implemented, that can display more information, and can output to the terminal, allowing the source of allocations to more easily be determined. Also, `Rprofmem` is now always accessible (not requiring the `--enable-memory-profiling` configuration option). Its overhead when not in use is negligible.  
The new version allows records of memory allocation to be output to the terminal, where their position relative to other output can be informative (this is the default for the new `Rprofmemt` variant). More identifying information, including type, number of elements, and hexadecimal address, can also be output. For more details on these and other changes, see `help(Rprofmem)`.
- A new primitive function, `pnamedcnt`, has been added, that prints the `NAMED-CNT/NAMED` count for an R object, which is helpful in tracking when objects will have to be duplicated. For details, see `help(pnamedcnt)`.
- The `tracemem` function is defunct. What exactly it was supposed to do in R-2.15.0 was unclear, and optimizations in pqR make it even less clear what it should do. The bit in object headers that was used to implement it has been put to a better use in pqR. The `--enable-memory-profiling` configuration option used to enable it no longer exists.  
The `retracemem` function remains for compatibility (doing nothing). The `Rprofmemt` and `pnamedcnt` functions described above provide alternative ways of gaining insight into memory allocation behaviour.
- Some options that can be set by arguments to the R command can now also be set with environment variables, specifically, the values of `R_DEBUGGER`, `R_DEBUGGER_ARGS`, and `R_HELPERS` give the default when `--debugger`, `--debugger-args`, and `--helpers` are not specified on the command line. This fea-

ture is useful when using a shell file or Makefile that contains R commands that one would rather not have to modify.

## INSTALLATION AND TESTING:

- The procedure for compiling and installing from source is largely unchanged from R-2.15.0. In particular, the final result is a program called "R", not "pqR", though of course you can provide a link to it called "pqR". Note that (as for R-2.15.0) it is not necessary to do an "install" after "make" — one can just run bin/R in the directory where you did "make". This may be convenient if you wish to try out pqR along with your current version of R.
- Testing of pqR has so far been done only on Linux/Unix systems, not on Windows or Mac systems. There is no specific reason to believe that it will not work on Windows or Mac systems, but until tests have been done, trying to use it on these systems is not recommended. (However, some users have reported that pqR can be built on Mac systems, as long as a C compiler fully supporting OpenMP is used, or the `--disable-helper-threads` configuration option is used.)
- This release contains the versions of the standard and recommended packages that were released with R-2.15.0. Newer versions may or may not be compatible (same as for R-2.15.0).
- It is intended that this release will be fully compatible with R-2.15.0, but you will need to recompile any packages (other than those with only R code) that you had installed for R-2.15.0, and any other C code you use with R, since the format of internal data structures has changed (see below).
- New configuration options relating to helper threads and to matrix multiplication now exist. For details, see `doc/R-admin.html` (or `R-admin.pdf`), or run `./configure --help`.  
In particular, the `--disable-helper-threads` option to configure will remove support for helper threads. Use of this option is advised if you know that multiple processors or processor cores will not be available, or if you know that the C compiler used does not support OpenMP 3.0 or 3.1 (which is used in the implementation of the helpers package).
- Including `-DENABLE_ISNAN_TRICK` in `CFLAGS` will speed up checks for NA and NaN on machines on which it works. It works on Intel processors (verified both empirically and by consulting Intel documentation). It does not work on SPARC machines.
- The `--enable-memory-profiling` option to configure no longer exists. In pqR, the `Rprofmem` function is always enabled, and the `tracemem` function is defunct. (See discussion above.)
- When installing from source, the output of configure now displays whether standard and recommended packages will be byte compiled.
- The tests of random number generation run with `make check-all` now set the random number seed explicitly. Previously, the random number seed was set from the time and process ID, with the result that these tests would occasionally fail non-deterministically, when by chance one of the p-values obtained was below the threshold used. (Any such failure should now occur consistently, rather than appearing to be due to a non-deterministic bug.)
- Note that (as in R-2.15.0) the output of `make check-all` for the boot package includes many warning messages regarding a non-integer argument, and when byte



compilation is enabled, these messages identify the wrong function call as the source. This appears to have no wider implications, and can be ignored.

- Testing of the "xz" compression method is now done with `try`, so that failure will be tolerated on machines that don't have enough memory for these tests.
- The details of how `valgrind` is used have changed. See the source file `'memory.c'`.

#### INTERNAL STRUCTURES AND APPLICATION PROGRAM INTERFACE:

- The internal structure of an object has changed, in ways that should be compatible with R-2.15.0, but which do require re-compilation. The flags in the object header for `DEBUG`, `RSTEP`, and `TRACE` now exist only for non-vector objects, which is sufficient for their present use (now that `tracemem` is defunct).
- The sizes of objects have changed in some cases (though not most). For a 32-bit configuration, the size of a cons cell increases from 28 bytes to 32 bytes; for a 64-bit configuration, the size of a cons cell remains at 56 bytes. For a 32-bit configuration, the size of a vector of one double remains at 32 bytes; for a 64-bit configuration (with 8-byte alignment), the size of a vector of one double remains at 48 bytes.
- Note that the actual amount of memory occupied by an object depends on the set of node classes defined (which may be tuned). There is no longer a separate node class for cons cells and zero-length vectors (as in R-2.15.0) — instead, cons cells share a node class with whatever vectors also fit in that node class.
- The old two-bit `NAMED` field of an object is now a three-bit `NAMEDCNT` field, to allow for a better attempt at reference counting. Versions of the `NAMED` and `SET_NAMED` macros are still defined for compatibility. See the R-ints manual for details.
- Setting the length of a vector to something less than its allocated length using `SETLENGTH` is deprecated. The `LENGTH` field is used for memory allocation tracking by the garbage collector (as is also the case in R-2.15.0), so setting it to the wrong value may cause problems. (Setting the length to more than the allocated length is of course even worse.)

#### PERFORMANCE IMPROVEMENTS:

- Many detailed improvements have been made that reduce general interpretive overhead and speed up particular functions. Only some of these improvements are noted below.
- Numerical computations can now be performed in parallel with each other and with interpretation of R code, by using "helper threads", on machines with multiple processors or multiple processor cores. When the output of one such computation is used as the input to another computation, these computations can often be done in parallel, with the output of one task being "pipelined" to the other task. Note that these parallel execution facilities do not require any changes to user code — only that helper threads be enabled with the `--helpers` option to the command starting `pqR`. See `help(helpers)` for details.

However, helper threads are not used for operations that are done within the interpreter for byte-compiled code or that are done in primitive functions invoked by the byte-code interpreter.

This facility is still undergoing rapid development. Additional documentation on which operations may be done in parallel will be forthcoming.

- A better attempt at counting how many "names" an object has is now made, which reduces how often objects are duplicated unnecessarily. This change is ongoing, with further improvements and documentation forthcoming.
- Several primitive functions that can generate integer sequences — `seq.int`, `seq_len`, and `seq_along` — will now sometimes not generate an actual sequence, but rather just a description of its start and end points. This is not visible to users, but is used to speed up several operations.

In particular, "for" loops such as `for (i in 1:1000000) ...` are now done without actually allocating a vector to hold the sequence. This saves both space and time. Also, a subscript such as `101:200` for a vector or as the first subscript for a matrix is now (often) handled without actually creating a vector of indexes, saving both time and space.

However, the above performance improvements are not effective in compiled code.

- Matrix multiplications with the `%*%` operator are now much faster when the operation is a vector dot product, a vector-matrix product, a matrix-vector product, or more generally when the sum of the numbers of rows and columns in the result is not much less than their product. This improvement results from the elimination of a costly check for NA/NaN elements in the operands before doing the multiply. There is no need for this check if the supplied BLAS is used. If a BLAS that does not properly handle NaN is supplied, the `%*%` operator will still handle NaN properly if the new library of matrix multiply routines is used for `%*%` instead of the BLAS. See the next two items for more relevant details.
- A new library of matrix multiply routines is provided, which is guaranteed to handle NA/NaN correctly, and which supports pipelined computation with helper threads. Whether this library or the BLAS routines are used for `%*%` is controlled by the `mat_mult_with_BLAS` option. The default is to not use the BLAS, but the `--enable-mat-mult-with-BLAS-by-default` configuration option will change this. See `help("%*%")` for details.
- The BLAS routines supplied with R were modified to improve the performance of the routines `DGEMM` (matrix-matrix multiply) and `DGEMV` (matrix-vector multiply). Also, proper propagation of NaN, Inf, etc. is now always done in these routines. This speeds up the `%*%` operator in R, when the supplied BLAS is used for matrix multiplications, and speeds up other matrix operations that call these BLAS routines, if the BLAS used is the one supplied.
- The low-level routines for generation of uniform random numbers have been improved. (These routines are also used for higher-level functions such as `runif`.) The previous code copied the seed back and forth to `.Random.seed` for every call of a random number generation function, which is rather time consuming given that for the default generator `.Random.seed` is 625 integers long. It also allocated new space for `.Random.seed` every time. Now, `.Random.seed` is used without copying, except when the generator is user-supplied. The previous code had imposed an unnecessary limit on the length of a seed for a user-supplied random number generator, which has now been removed.
- The `any` and `all` primitives have been substantially sped up for large vectors. Also, expressions such as `all(v>0)` and `any(is.na(v))`, where `v` is a real vector, avoid computing and storing a logical vector, instead computing the result of `any` or `all` without this intermediate, looking at only as much of `v` as is needed to determine the result. However, this improvement is not effective in compiled code.

- When `sum` is applied to many mathematical functions of one vector argument, for example `sum(log(v))`, the sum is performed as the function is computed, without a vector being allocated to hold the function values. However, this improvement is not effective in compiled code.
- The handling of power operations has been improved (primarily for powers of reals, but slightly affecting powers of integers too). In particular, scalar powers of 2, 1, 0, and -1, are handled specially to avoid general power operations in these cases.
- Extending lists and character vectors by assigning to an index past the end, and deleting list items by assigning NULL have been sped up substantially.
- The speed of the transpose (`t`) function has been improved, when applied to real, integer, and logical matrices.
- The `cbind` and `rbind` functions have been greatly sped up for large objects.
- The `c` and `unlist` functions have been sped up by a bit in simple cases, and by a lot in some situations involving names.
- The `matrix` function has been greatly sped up, in many cases.
- Extraction of subsets of vectors or matrices (eg, `v[100:200]` or `M[1:100,101:110]`) has been sped up substantially.
- Logical operations and relational operators have been sped up in simple cases. Relational operators have also been substantially sped up for long vectors.
- Access via the `$` operator to lists, pairlists, and environments has been sped up.
- Existing code for handling special cases of `"["` in which there is only one scalar index was replaced by cleaner code that handles more cases. The old code handled only integer and real vectors, and only positive indexes. The new code handles all atomic vectors (logical, integer, real, complex, raw, and string), and positive or negative indexes that are not out of bounds.
- Many unary and binary primitive functions are now usually called using a faster internal interface that does not allocate nodes for a pairlist of evaluated arguments. This change substantially speeds up some programs.
- Lookup of some builtin/special function symbols (eg, `"+"` and `"if"`) has been sped up by allowing fast bypass of non-global environments that do not contain (and have never contained) one of these symbols.
- Some binary and unary arithmetic operations have been sped up by, when possible, using the space holding one of the operands to hold the result, rather than allocating new space. Though primarily a speed improvement, for very long vectors avoiding this allocation could avoid running out of space.
- Some speedup has been obtained by using new internal C functions for performing exact or partial string matches in the interpreter.

#### BUG FIXES:

- The `"debug"` facility has been fixed. Its behaviour for `if`, `while`, `repeat`, and for statements when the inner statement was or was not one with curly brackets had made no sense. The fixed behaviour is now documented in `help(debug)`. (I reported this bug and how to fix it to the R Core Team in July 2012, but the bug is still present in R-3.0.1, released May 2013.)
- Fixed a bug in `sum`, where overflow is allowed (and not detected) where overflow can actually be avoided. For example:

```
> v<-c(3L,1000000000L:1010000000L,-(1000000000L:1010000000L))
> sum(v)
[1] 4629
```

Also fixed a related bug in `mean`, applied to an integer vector, which would arise only on a system where a long double is no bigger than a double.

- Fixed `diag` so that it returns a matrix when passed a list of elements to put on the diagonal.
- Fixed a bug that could lead to mis-identification of the direction of stack growth on a non-Windows system, causing stack overflow to not be detected, and a segmentation fault to occur. (I also reported this bug and how to fix it to the R Core Team, who included a fix in R-2.15.2.)
- Fixed a bug where, for example, `log(base=4)` returned the natural log of 4, rather than signalling an error.
- The documentation on what `MARGIN` arguments are allowed for `apply` has been clarified, and checks for validity added. The call

```
> apply(array(1:24,c(2,3,4)),-3,sum)
```

now produces correct results (the same as when `MARGIN` is 1:2).

- Fixed a bug in which `Im(matrix(complex(0),3,4))` returned a matrix of zero elements rather than a matrix of NA elements.
- Fixed a bug where more than six warning messages at startup would overwrite random memory, causing garbage output and perhaps arbitrarily bizarre behaviour.
- Fixed a bug where `LC_PAPER` was not correctly set at startup.
- Fixed `gc.time`, which was producing grossly incorrect values for user and system time.
- Now check for bad arguments for `.rowSums`, `.colSums`, `.rowMeans`, and `.colMeans` (would previously segfault if `n*p` too big).
- Fixed a bug where excess warning messages may be produced on conversion to `RAW`. For instance:

```
> as.raw(1e40)
[1] 00
Warning messages:
1: NAs introduced by coercion
2: out-of-range values treated as 0 in coercion to raw
```

Now, only the second warning message is produced.

- A bug has been fixed in which `rbind` would not handle non-vector objects such as function closures, whereas `cbind` did handle them, and both were documented to do so.
- Fixed a bug in `numeric_deriv` in `stats/src/nls.c`, where it was not duplicating when it should, as illustrated below:

```
> x <- 5; y <- 2; f <- function (y) x
> numericDeriv(f(y),"y")
[1] 5
attr(,"gradient")
[1,] 0
> x
[1] 5
```

```
attr("gradient")
      [,1]
[1,]      0
```

- Fixed a bug in `vapply` illustrated by the following:

```
X<-list(456)
f<-function(a)X
A<-list(1,2)
B<-vapply(A,f,list(0))
print(B)
X[[1]][1]<-444
print(B)
```

After the fix, the values in `B` are no longer changed by the assignment to `X`. Similar bugs in `mapply`, `eapply`, and `rapply` have also been fixed. I reported these bugs to `r-devel`, and (different) fixes are in `R-3.0.0` and later versions.

- Fixed a bug in `rep.int` illustrated by the following:

```
a<-list(1,2)
b<-rep.int(a,c(2,2))
b[[1]][1]<-9
print(a[[1]])
```

- Fixed a bug in `mget`, illustrated by the following code:

```
a <- numeric(1)
x <- mget("a",as.environment(1))
print(x)
a[1] <- 9
print(x)
```

- Fixed bugs that the R Core Team fixed (differently) for `R-2.15.3`, illustrated by the following:

```
a <- list(c(1,2),c(3,4))
b <- list(1,2,3)
b[2:3] <- a
b[[2]][2] <- 99
print(a[[1]][2])
```

```
a <- list(1+1,1+1)
b <- list(1,1,1,1)
b[1:4] <- a
b[[1]][1] <- 1
print(b[2:4])
```

- Fixed a bug illustrated by the following:

```
> library(compiler)
> foo <- function(x,y) UseMethod("foo")
> foo.numeric <- function(x,y) "numeric"
> foo.default <- function(x,y) "default"
> testi <- function () foo(x=NULL,2)
> testc <- cmpfun (function () foo(x=NULL,2))
> testi()
[1] "default"
```

```
> testc()
[1] "numeric"
```

- Fixed several bugs that produced wrong results such as the following:

```
a<-list(c(1,2),c(3,4),c(5,6))
b<-a[2:3]
a[[2]][2]<-9
print(b[[1]][2])
```

I reported this to r-devel, and a (different) fix is in R-3.0.0 and later versions.

- Fixed bugs reported on r-devel by Justin Talbot, Jan 2013 (also fixed, differently, in R-2.15.3), illustrated by the following:

```
a <- list(1)
b <- (a[[1]] <- a)
print(b)
a <- list(x=1)
b <- (a$x <- a)
print(b)
```

- Fixed `svd` so that it will not return a list with `NULL` elements. This matches the behaviour of `La.svd`.
- Fixed (by a kludge, not a proper fix) a bug in the "tre" package for regular expression matching (eg, in `sub`), which shows up when `WCHAR_MAX` doesn't fit in an "int". The kludge reduces `WCHAR_MAX` to fit, but really the "int" variables ought to be bigger. (This problem showed up on a Raspberry Pi running Raspbian.)
- Fixed a minor error-reporting bug with `(1:2):integer(0)` and similar expressions.
- Fixed a small error-reporting bug with `"$"`, illustrated by the following output:

```
> options(warnPartialMatchDollar=TRUE)
> pl <- pairlist(abc=1,def=2)
> pl$ab
[1] 1
Warning message:
In pl$ab : partial match of 'ab' to ''
```

- Fixed documentation error in R-admin regarding the `--disable-byte-compiled-packages` configuration option, and changed the `DESCRIPTION` file for the recommended `mgcv` package to respect this option.
- Fixed a bug reported to R Core (PR 15363, 2013-006-26) that also existed in `pqR`-2013-06-20. This bug sometimes caused memory expansion when many complex assignments or removals were done in the global environment.