NEWS for pqR version 2.15.0 (2013-06-28)

NEWS *pqR News*

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.
- 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 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 feature 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 that 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 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 `rnorm`.)
  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 .rowMeans (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]
 [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 long 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 but 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.