Recently a query by Kevin Ummel on the R-help mailing list prompted a discussion of a problem that boils down to comparing the elements of two numeric vectors, `x` and `y`, and determining for each element in one vector the number of elements in the second vector that are less than or equal to it.

There are various ways of doing this. The original poster used

```
> f1 <- function(x, y)
+     sapply(x, function(i) length(which(y < i)))
```

Richard Heiberger and Marc Swartz both suggested

```
> f2 <- function(x, y)
+     colSums(outer(y, x, '<'))
```

Gustavo Carvalho suggested the equivalent of

```
> f3 <- Vectorize(function(x, y) sum(y < x), "x")
```

and Bill Dunlap, drawing on his encyclopedic knowledge of S-PLUS and R functions, noted that this operation was essential what is done in R's findInterval function which uses compiled code implementing a binary search.

```
> f4 <- function(x, y) length(y) - findInterval(-x, rev(-sort(y)))
```

For large vectors `x` and `y`, Bill's version is much faster than any of the other suggestions which involve comparing each element of `x` to each elements of `y`. Interestingly, the second version (`f2`), which was suggested by two experience R users, can become deadly slow on moderate sized vectors, because of the way that the `outer` function is implemented.

Even with moderate sized vectors

```
> set.seed(1)
> x <- rnorm(5000)
> y <- rnorm(20000)
> system.time(a1 <- f1(x, y))
   user  system elapsed
  3.194   0.029   3.223

> system.time(a2 <- f2(x, y))
   user  system elapsed
 12.567   1.645  14.215

> system.time(a3 <- f3(x, y))
   user  system elapsed
  2.568   0.183   2.753

> system.time(a4 <- f4(x, y))
   user  system elapsed
  0.007   0.000   0.007

> all.equal(a1, a2)
[1] TRUE

> all.equal(a1, a3)
[1] TRUE

> all.equal(a1, a4)
[1] TRUE
```

```
> benchmark(f1(x,y), f2(x,y), f3(x,y), f4(x,y),
+           columns=c("test","elapsed","relative"),
+           order="relative", replications=10L)
      test elapsed  relative
4 f4(x, y)   0.066    1.0000
3 f3(x, y)  25.881  392.1364
1 f1(x, y)  28.919  438.1667
2 f2(x, y) 136.875 2073.8636
>
```

We will eliminate all but Bill Dunlap's method on this evidence and change the rules a bit. The question posed by Sunduz Keles regarded p-values from a test sample relative to a larger reference sample

(From here you can continue with the description on the R-help and Rcpp-Devel postings in which I included benchmark timings of various versions.)