How to preserve attributes of a data.frame

In the following we have two expressions in which the attributes of a data.frame will be lost.

```
dat <- data.frame(x = 1:10, y = "")
attr(dat, "newAttr") <- 5

# works:
attributes(dat[1:5,])
```

```
## $names
## [1] "x" "y"
##
## $newAttr
## [1] 5
##
## $row.names
## [1] 1 2 3 4 5
##
## $class
## [1] "data.frame"
```

```
# dplyr is evil
attributes(dplyr::filter(dat, x %in% 1:5))
```

```
## $class
## [1] "data.frame"
##
## $row.names
## [1] 1 2 3 4 5
##
## $names
## [1] "x" "y"
```

Can we avoid this by using a S4 data.frame?

```
S4df <- setClass(
    "S4df",
    contains = "data.frame",
    slots = list("attributes" = "list")
    )

s4dat <- S4df(dat, attributes = attributes(dat))
```

```r
attributes(dplyr::filter(s4dat, x %in% 1:5)) # does not work...
```

```
## Error in eval(expr, envir, enclos): could not convert using R function : as.data.fram
```

Okay, functional programming. . .

```r
preserve_attributes <- function(fun) {
    force(fun)
    function(dat) {
        attOfX <- attributes(dat)
        res <- fun(dat)
        attOfRes <- attributes(res)
        attToPreserve <- names(attOfX)[!(names(attOfX) %in% names(attOfRes))]
        attributes(res) <- c(attributes(res), attributes(dat)[attToPreserve])
        res
    }
}

myFilter <- preserve_attributes(
    functional::CurryL(
        dplyr::filter, "..." = x %in% 1:5
        )
    )

attributes(myFilter(dat))
```

```
## $class
## [1] "data.frame"
##
## $row.names
## [1] 1 2 3 4 5
##
## $names
## [1] "x" "y"
##
## $newAttr
## [1] 5
```