This little useless-useful R function is not that useless after-all. Many of the R programmers have had some issues with namespaces in R environment and lost some time debugging, that turned out to be due to variable name or function name.

Let's make a simple example:

```
c <- 100
a <- c(a,c)

a
```

So it looks like two variables names "a" and "c". YeeaNo. "c" is reserved word (means combine) and can easily be used also as a named variable. Which is a programmers fault. But the interesting part is when you want to see what is stored in variable a. To your surprise, the value of variable and the function itself. So, caution when using reserved words for variables names. Another problems are namespaces and same function names among different packages. And you want to know about these incidents, right?

```
> a
[[1]]
function (..., .noWS = NULL)
{
    validateNoWS(.noWS)
    contents <- dots_list(...)
    tag(tagname, contents, .noWS = .noWS)
}
<bytecode: 0x7fba517e4500>
<environment: 0x7fba61393378>

[[2]]
[1] 100

[[3]]
[1] 100

[[4]]
[1] 100

>
```

So the following function does just that; if goes through the list of functions for all the installed packages i particular environment and finds duplicates. Giving the R programmer the insights where to be cautious and refer to scope and namespace.

```
funkyFun <- function(){
     libnames <- installed.packages()[,1]
     #exclude packages:
     libnames <- libnames[ !(libnames %in% c("rJava","RPostgreSQL","
XLConnect","xlsx","xlsxjars")) ]
     df <- data.frame(packname = NULL, funkyName = c(NULL,NULL))
     #for (i in 1:50){
     for (i in 1:length(libnames)){
        com <- paste0("require(", libnames[i],")")
        eval(parse(text= com))
        str <- paste0("package:", libnames[i])
        funk <- (ls(str))
        if (length(funk)==0){
          funk <- ifelse((length(funk)==0)==TRUE, "funkyFun", funk)
        }
        da <- cbind(libnames[i], funk)
        df <- rbind(df, da)

     }
     no_freq <- data.frame(table(df$funk))
     all_duplicated_functions  <- no_freq[no_freq$Freq > 1,]
     all_duplicated_functions_per_package <- df[df$funk %in%
no_freq$Var1[no_freq$Freq > 1],]
     return(all_duplicated_functions_per_package)
}
```

Results of the function is somewhat interesting. In my case, with installed little over 300 packages (yes, it's not that much), I have managed to find at least 20 functions that share same name and each of them is present in at least two (or more) packages.

Function	Frequency	available in Packages
get	2	base, config
logit	2	boot, car
match	2	base, bit64
merge	2	base, config
order	2	base, bit64
rank	2	base, bit64
Recall	2	base, caret
setattr	2	bit, data.table
show	2	coin, colorspace
symbol	2	cli, clisymbols

And this is just an excerpt from much larger set of duplicate function names among the packages. And also gives you the idea of potentially conflicting packages, as on the example above, caret and base for function Recall. When loading caret (to already existing base function), be cautious when calling such

functions.