A question was asked in the NHS-R Slack channel a few weeks back. It was one of those strangely innocuous, but fiddly ones. A dplyr solution was offered, and accepted, but someone has to be the annoying person who offers a data.table solution, and that person is me.

A badge I wear with honour, it has to be said.

Here is the example data:

```
demo <- structure(list(System = c("A", "A"),
                SpecialtyDescription = c("B","B"),
                oneService = c(0, 1),
                multipleService = c(1, 0),
                patients = c(10L,10L)),
           class = c("tbl_df", "tbl", "data.frame"),
           row.names = c(NA,-2L))

demo
```

```
##   System SpecialtyDescription oneService multipleService patients
## 1      A                    B          0               1       10
## 2      A                    B          1               0       10
```

The question

How do we summarise this so there is only one line, containing the values that equal 1?

The dplyr answer :

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
demo %>% group_by(System,
            SpecialtyDescription) %>%
  summarise(oneService = max(oneService, na.rm = TRUE),
             multipleService = max(multipleService, na.rm = TRUE))
```

```
## `summarise()` regrouping output by 'System' (override with `.groups`
argument)
```

```
## # A tibble: 1 x 4
## # Groups:   System [1]
##   System SpecialtyDescription oneService multipleService
##
## 1 A      B                             1               1
```

Not bad. How else could this be achieved?

Enter data.table

```
# use data.table, copy the original dataframe and make sure it's a data.table
library(data.table)
```

```
##
## Attaching package: 'data.table'
```

```
## The following objects are masked from 'package:dplyr':
##
##     between, first, last
```

```
DT <- copy(demo)
setDT(DT)
```

Note. I could just have done

```
setDT(demo)
```

But I prefer to make a copy, so that data.table doesn't update the original by reference.
That can be hard to get used to at first, and I may write a post about that.
Or do a video. Haven't decided if 2020 is ready for that yet..

The not so good approach

My first attempt, which was very much evidence of 'late Friday afternoon brain':

```
unique(DT[,`:=`(oneService = max(oneService,na.rm = TRUE), multipleService =
max(multipleService, na.rm = TRUE)),
by =c('System','SpecialtyDescription','patients')][])
```

```
##    System SpecialtyDescription oneService multipleService patients
## 1:      A                    B          1               1       10
```

Well, it worked (with a lot less hassle than some of the other attempts
;)) But it doesn't feel right, there's too much repetition, and that's
not the concise, data.table way.

The swish solution

```
DT[,lapply(.SD, max),.SDcols = c('oneService', 'multipleService'), by =
c('System','SpecialtyDescription')][]
```

```
##    System SpecialtyDescription oneService multipleService
## 1:      A                    B          1               1
```

Much better.
We only need to specify the function we want to apply once.
Incidentally, I'd managed to avoid using lapply for years until I started using data.table.
I don't think it does any harm to know some base R, or non tidyverse alternatives.
At some point, you have to take the stabilisers off..

You will also notice I left the 'patients' column off this time, as that is what the dplyr solution also does.

Alternatively :

```
cols_to_summarise <- c('oneService', 'multipleService')
by_cols <- c('System','SpecialtyDescription')
DT[,lapply(.SD, max),.SDcols = cols_to_summarise, by = by_cols]
```

```
##    System SpecialtyDescription oneService multipleService
## 1:      A                    B          1               1
```

This is handy if the columns are likely to be referred to later on in an analysis pipeline. You define them once, up front, and refer to them throughout. Although, in that case, I'd have used a snappier title than 'cols_to_summarise'.

Anyway, fellow NHS-R folk, who I love, are now given fair warning that I will continue to be the annoying data.table guy, because at some point, a really tricky problems will arise, and data.table is a great tool to have available. You should learn it too.